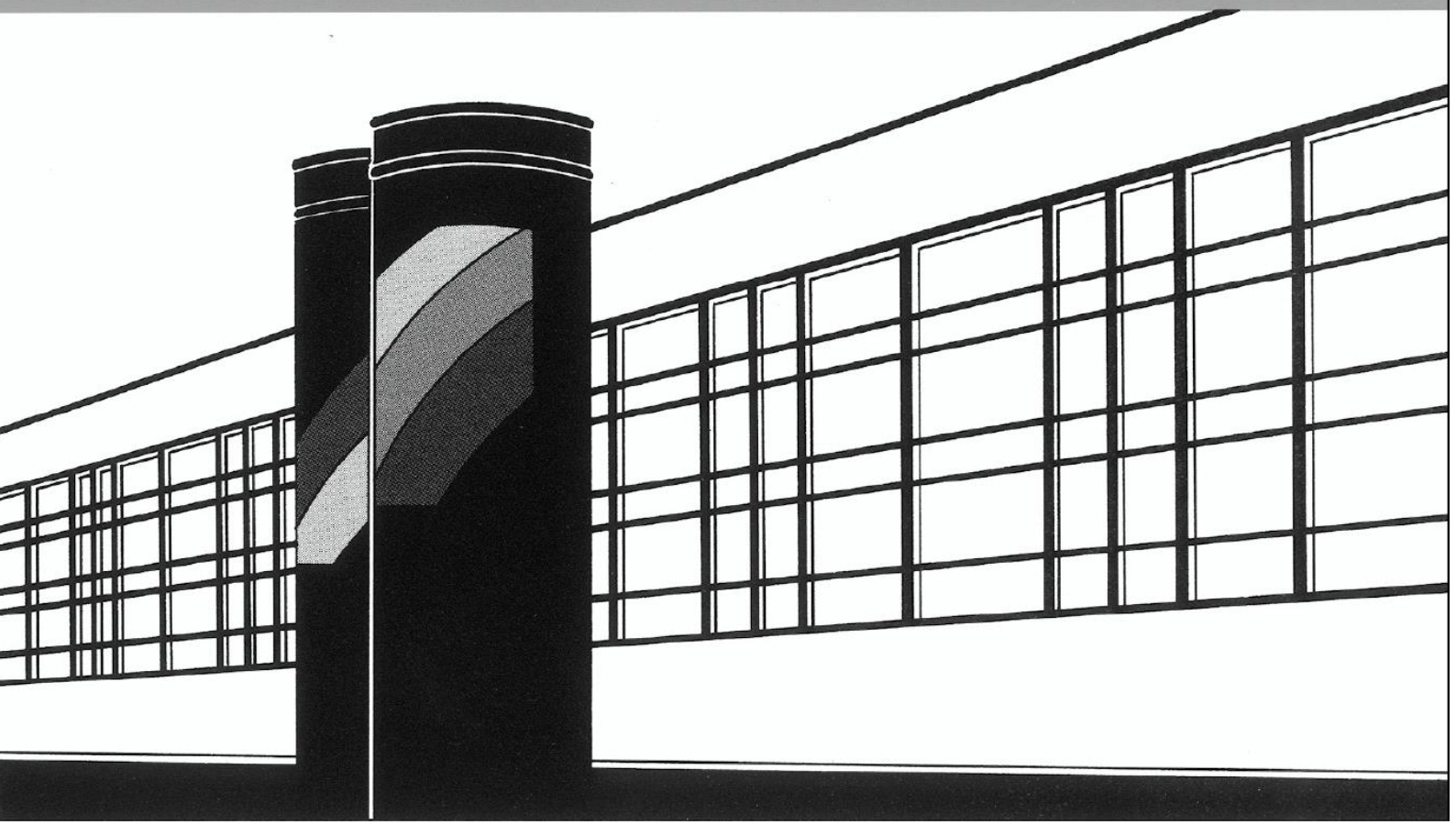
Universität Stuttgart

Institut für Wasser- und Umweltsystemmodellierung

Mitteilungen



Heft 300 Timothy Praditia

Physics-informed Neural Networks for Learning Dynamic, Distributed and Uncertain Systems

Physics-informed Neural Networks for Learning Dynamic, Distributed and Uncertain Systems

von der Fakultät Bau- und Umweltingenieurwissenschaften
der Universität Stuttgart und dem Stuttgart Center for Simulation Science
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von
Timothy Praditia
aus Jakarta, Indonesien

Hauptberichter:	Prof. Dr.-Ing. Wolfgang Nowak
Mitberichter:	apl. Prof. Dr.-Ing Sergey Oladyshkin
	Prof. Dr. Mathias Niepert
	Prof. Dr. Martin V. Butz
Tag der mündlichen Prüfung:	08. Mai 2023

Institut für Wasser- und Umweltsystemmodellierung
der Universität Stuttgart
2023

Heft 300

Physics-informed Neural Networks for Learning Dynamic, Distributed and Uncertain Systems

von
Dr.-Ing.
Timothy Praditia

Eigenverlag des Instituts für Wasser- und Umweltsystemmodellierung
der Universität Stuttgart

D93 Physics-informed Neural Networks for Learning Dynamic, Distributed and Uncertain Systems

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://www.d-nb.de abrufbar

Praditia, Timothy:
Physics-informed Neural Networks for Learning Dynamic, Distributed and Uncertain Systems, Universität Stuttgart. - Stuttgart: Institut für Wasser- und Umweltsystemmodellierung, 2023

(Mitteilungen Institut für Wasser- und Umweltsystemmodellierung, Universität Stuttgart: H. 300)
Zugl.: Stuttgart, Univ., Diss., 2023
ISBN 978-3-910293-04-5
NE: Institut für Wasser- und Umweltsystemmodellierung <Stuttgart>: Mitteilungen

Herausgegeben 2023 vom Eigenverlag des Instituts für Wasser- und Umweltsystem-modellierung

Druck: DCC Kästl e.K., Ostfildern

Acknowledgements

I would like to thank everyone who contributed directly and indirectly to the completion of this thesis. I could not have undertaken this journey without Wolfgang Nowak and Sergey Oladyshkin, who took a chance on hiring me based solely on a recommendation and an online chat. I would like to especially thank both of you for all the scientific support and for being great discussion partners. Special thanks also go to Rainer Helmig, who introduced me to Wolfgang in the first place. I am also extremely grateful to Ute Coquet and Astrid Lemp, for their extreme patience and great help with all my administrative matters. I would also like to thank the German Research Foundation (Deutsche Forschungsgemeinschaft), which financed this project within the framework of the Cluster of Excellence Simulation Technology.

I would like to express my sincere thanks to my colleagues in the LS3 group, especially to Ishani Banerjee, Ana González-Nicolás, Aline Schäfer Rodrigues Silva, Sebastian Reuschen, Sinan Xiao, Tim Brünnette, Patrick Martin, and Maria Fernanda Morales Oreamuno, for all the random conversations, the activities that we did together, and the comforts that you brought to the group. Special thanks go to Nils Wildt, for the thought-provoking discussions and great help in proofreading this thesis. I would also like to extend my gratitude to my co-authors, with whom I had the pleasure of collaborating: Matthias Karlbauer, Sebastian Otte, Coşku Can Horuz, and Martin V. Butz. I thank all of you for the fruitful and productive collaboration.

Finally, words cannot express my gratitude to my parents, who never gave up on me and ensured that I had access to proper education, even during tough times. I am also thankful for all my friends, whose names I cannot mention one by one, for the emotional support that you have provided. Last but not least, I would like to express my deepest appreciation to Dinar Suryandari, for being understanding and supportive, and for being there for me all the time.

Contents

List of Figures

Nomenclature

Selected Acronyms

ADR	Advection-Diffusion-Reaction
ANN	Artificial Neural Network
BNN	Bayesian Neural Network
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
CPU	Central Processing Unit
DISTANA	Distributed Spatio-Temporal Artificial Neural Network Architecture
FDM	Finite Difference Method
FEM	Finite Element Method
FINN	Finite Volume Neural Network
FNO	Fourier Neural Operator
FVM	Finite Volume Method
GPU	Graphics Processing Unit

GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MALA	Metropolis-Adjusted Langevin Algorithm
MCMC	Markov Chain Monte Carlo
MH	Metropolis-Hastings
MPNN	Message Passing Neural Network
NARX	Nonlinear Autoregressive Network with Exogeneous Input
ODE	Ordinary Differential Equation
P	Parallel
PDE	Partial Differential Equation
PINN	Physics Informed Neural Network
ResNet	Residual Network
RNN	Recurrent Neural Network
SP	Series-Parallel
TCES	Thermochemical Energy Storage
TCN	Temporal Convolutional Network

Symbols

Δt	Time step size
Δx	Length of the control volume
ϵ	Random noise

$\hat{n}$	Normal vector pointing out of the control volume surface
$\hat{u}$	Prediction of the variable of interest
$\mathbb{E}$	Expected value
$\mathcal{F}$	Fourier transform
$\mathcal{F}^{-1}$	Inverse Fourier transform
$\mathcal{R}$	Linear transform
ν_i	Control volume with index i
Ω	Control volume surface
$\sigma(\cdot)$	Sigmoid function
$\tanh(\cdot)$	Hyperbolic tangent function
τ	Markov Chain Monte Carlo step size
$a(\cdot)$	Nonlinear activation function
A_i	Cross-sectional surface area of control volume i
a_t	Neural ODE adjoint state at time step t
c_t	LSTM cell state
D	Diffusion coefficient
$D(\cdot)$	Decoder
d_u	NARX feedback delay
D_{KL}	Kullback-Leibler divergence
$E(\cdot)$	Encoder

$f(u,t)$	Rate of change as a function of u and t
f_t	LSTM forget gate
h_t	Input at time step t
i_t	LSTM input gate
k	Convolutional filter size
K_t	Kalman gain
$L(\hat{u},u)$	Loss function
N_d	Number of data points
N_f	Number of physics-based loss collocation points
N_x	Number of control volumes
$N_{w,l}$	Number of hidden nodes in layer l
$p(w)$	Prior distribution of the ANN parameters w
pk	DISTANA prediction kernel
q	Source/sink term
r_t	PhyDNet residual
t	Time
tk	DISTANA transition kernel
u	Variable of interest
u_t''	Second derivative of u with respect to time t
u_t'	First derivative of u with respect to time t

v	Advective velocity
$w_{i,j}^{l}$	ANN parameter that connects node j from layer $l-1$ to node i in layer l
x	Spatial coordinate

Abstract

Scientific models play an important role in many technical inventions to facilitate daily human activities. We use them to assist us in simple decision making such as deciding what type of clothing we should wear using the weather forecast model, and also in complex problems such as assessing the environmental impact of industrial wastes. Existing scientific models, however, are imperfect due to our limited understanding of complex physical systems. Due to the rapid growth in computing power in recent years, there has been an increasing interest in applying data-driven modeling to improve upon current models and to fill in the missing scientific knowledge. Traditionally, these data-driven models require a significant amount of observation data, which is often challenging to obtain, especially from a natural system. To address this issue, prior physical knowledge has been included in the model design, resulting in so-called hybrid models. Although the idea of infusing physics with data seems sound, current state-of-the-art models have not found the ideal combination of both aspects, and the application to real-world data has been lacking.

To bridge this gap, three research questions are formulated:

1. How can prior physical knowledge be adopted to design a consistent and reliable hybrid model for dynamic systems?

2. How can prior physical and numerical knowledge be adopted to design a consistent and reliable hybrid model for dynamic and spatially distributed systems?

3. How can the hybrid model learn about its own total (predictive) uncertainty in a computationally effective manner, so that it is appropriate for real-world applications or could facilitate scientific hypothesis testing?

The overall goal is, with these questions answered, to contribute to more consistent approaches for scientific inquiry through hybrid models.

The first contribution of this thesis addresses the first research question by proposing a modeling framework for a dynamic system, in the form of a Thermochemical Energy Storage device. A Nonlinear Autoregressive Network with Exogeneous Input (NARX) model is trained recurrently with multiple time lags to capture the temporal dependency and the long-term dynamics of the system. During training, the model is penalized when it violates established physical laws, such as mass and energy conservation. As a result, the model produces accurate and physically plausible predictions compared to models that are trained without physical regularization.

The second research question is addressed by the second contribution of this thesis, by designing a hybrid model that complements the Finite Volume Method (FVM) with the learning ability of Artificial Neural Networks (ANNs). The resulting model enables the learning of unknown closure/constitutive relationships in various advection-diffusion equations. This thesis shows that the proposed model outperforms state-of-the-art deep learning models by several orders of magnitude in accuracy, and it possesses excellent generalization ability.

Finally, the third contribution addresses the third research question, by investigating the performance of assorted uncertainty quantification methods on the hybrid model. As a demonstration, laboratory measurement data of a groundwater contaminant transport process is employed to train the model. Since the available training data is extremely scarce and noisy, uncertainty quantification methods are essential to produce a robust and trustworthy model. It is shown that a gradient-based Markov Chain Monte Carlo (MCMC) algorithm, namely the Barker proposal is the most suitable to quantify the uncertainty of the proposed model. Additionally, the hybrid model outperforms a calibrated physical model and provides appropriate predictive uncertainty to sufficiently explain the noisy measurement data.

With these contributions, this thesis proposes a robust hybrid modeling framework that is suitable for filling in missing scientific knowledge and lays the groundwork for a wider variety of complex real-world applications. Ultimately, the hope is for this work to inspire future studies that contribute to the continuous and mutual improvements of both scientific knowledge discovery and scientific model robustness.

Zusammenfassung

Wissenschaftliche Modelle spielen eine wichtige Rolle in vielen technischen Erfindungen, die den Menschen den Alltag erleichtern. Wir nutzen sie, um einfache Entscheidungen zu treffen, zum Beispiel die Wahl der richtigen Kleidung anhand eines Wettervorhersagemodells, aber auch bei komplexen Problemen wie der Bewertung der Umweltauswirkungen von Industrieabfällen. Bestehende wissenschaftliche Modelle sind jedoch aufgrund unseres begrenzten Verständnisses komplexer physikalischer Systeme unvollkommen. Aufgrund der raschen Zunahme der Rechenleistung in den letzten Jahren hat das Interesse an datengetriebenen Modellen zugenommen, um aktuelle Modelle zu verbessern und fehlende wissenschaftliche Erkenntnise zu ergänzen. Diese datengetriebenen Modelle erfordern traditionell eine beträchtliche Menge an Beobachtungsdaten, die oft schwer zu beschaffen sind, insbesondere bei natürlichen Systemen. Um dieses Problem zu lösen, wurde physikalisches Vorwissen in den Modellentwurf einbezogen, was zu sogenannten Hybridmodellen führte. Obwohl die Idee, die Physik mit Daten zu verbinden, vernünftig erscheint, fällt es aktuellen Modellen schwer, die ideale Kombination beider Aspekte zu finden. Deshalb findet hybride Modellierung noch keine breite Anwendung auf reale Daten.

Um diese Probleme zu lösen, werden drei Forschungsfragen formuliert:

1. Wie kann physikalisches Vorwissen genutzt werden, um ein konsistentes und zuverlässiges Hybridmodell für dynamische Systeme zu entwerfen?

2. Wie kann physikalisches Vorwissen genutzt werden, um ein konsistentes und zuverlässiges hybrides Modell für dynamische und verteilte Systeme zu entwerfen?

3. Wie kann das hybride Modell seine eigene gesamte Vorhersageunsicherheit auf eine rechnerisch effektive Weise erlernen, so dass es für reale Anwendungen geeignet ist?

Das Gesamtziel ist es, mit der Beantwortung dieser Fragen zu kohärenteren Ansätzen für wissenschaftliche Untersuchungen durch hybride Modelle beizutragen.

Der erste Beitrag dieser Arbeit befasst sich mit der ersten Forschungsfrage, wobei ein Modellierungsrahmen für ein dynamisches System in Form eines thermochemischen Energiespeichers betrachtet wird. Ein "Nonlinear Autoregressive Network with Exogeneous Input"(NARX) Model wird rekurrent mit mehreren Zeitverzögerungen trainiert, um die zeitliche Abhängigkeit und die langfristige Dynamik des Systems zu erfassen. Während des Trainings wird das Modell bestraft, wenn es gegen etablierte physikalische Gesetze verstößt, wie z. B. die Erhaltung von Masse und Energie. Infolgedessen liefert das Modell genaue und physikalisch plausible Vorhersagen im Vergleich zu Modellen, die ohne physikalische Regularisierung trainiert werden.

Die zweite Forschungsfrage wird durch den zweiten Beitrag dieser Arbeit beantwortet, in welchem ein hybrides Modell entwickelt wird, das die Finite-Volumen-Methode (FVM) mit der Lernfähigkeit künstlicher neuronaler Netze (ANN) ergänzt. Das daraus resultierende Modell ermöglicht das Lernen unbekannter Schließungs-/Konstitutivbeziehungen in verschiedenen Advektions-Diffusions-Gleichungen. Das vorgeschlagene Modell übertrifft die modernsten Deep-Learning-Modelle um mehrere Größenordnungen in der Genauigkeit und besitzt eine hervorragende Generalisierungsfähigkeit.

Der dritte Beitrag befasst sich schließlich mit der dritten Forschungsfrage, indem verschiedene Methoden zur Quantifizierung von Unsicherheiten im Hybridmodell untersucht. Zur Demonstration werden Labormessdaten über Grundwasserschadstofftransport zum Trainieren des Modells verwendet. Da die verfügbaren Trainingsdaten extrem knapp und verrauscht sind, sind Methoden zur Quantifizierung der Unsicherheit unerlässlich, um ein robustes und zuverlässiges Modell zu erstellen. Es wird gezeigt, dass ein gradientenbasierter Markov-Chain-Monte-Carlo-Algorithmus (MCMC), nämlich der Barker-Proposal, am besten geeignet ist, die Unsicherheit des vorgeschlagenen Modells zu quantifizieren. Darüber hinaus übertrifft das Hybridmodell ein kalibriertes physikalisches Modell und bietet eine angemessene Vorhersageunsicherheit, um die verrauschten Messdaten ausreichend zu erklären.

Mit diesen Beiträgen schlägt diese Arbeit einen robusten hybriden Modellierungsrahmen vor, der geeignet ist, fehlendes wissenschaftliches Wissen zu ergänzen und die Grundlage für eine breitere Palette komplexer Anwendungen in der realen Welt zu legen. Letztendlich hoffen wir, dass diese Arbeit künftige Studien anregt. Folglich wäre es

näher, eine kontinuierliche und gegenseitige Verbesserung sowohl der wissenschaftlichen Erkenntnisgewinnung als auch der Robustheit wissenschaftlicher Modelle zu erreichen.

1 Introduction

1.1 Background

Scientists have been striving to develop reliable scientific models to predict the behavior of natural systems and ultimately aid in strategic decision-making processes for solving various problems. Such problems include groundwater contamination, where the modeling of contaminant diffusion processes is essential to determine whether the groundwater is contaminated [e.g. Brown et al., 2012, Koch and Nowak, 2015, Nowak and Guthke, 2016]; climate change, where accurate climate models are required to quantify the potential effect of global warming and devise a robust strategy to minimize the catastrophe [e.g. Marchuk, 1974, Wetherald and Manabe, 2002, IPCC, 2013]; and earthquake mitigation, where reliable models are necessary to assess potential hazards, as well as to plan and design robust infrastructure [e.g. Woessner et al., 2015, Crowley et al., 2019, Allen et al., 2020].

However, despite centuries of scientific discovery and progress, complex physical systems are still poorly understood, especially at problem-relevant scales. At smaller scales, such as molecular and pore scales, where the physical processes are usually better understood, the computational cost is often prohibitive to model and simulate the systems. Additionally, upscaling the model from small scale to problem-relevant scale universally leads to inaccuracies due to missing or unknown governing laws across scales and heterogeneity of the small-scale systems [e.g. Bourg and Sposito, 2010, Molins and Knabner, 2019, Lavin et al., 2021].

Most of these problems have three common characteristics. First, they are dynamic, which means that the systems are time-dependent, and therefore, the system states evolve, depending on their values at an earlier point in time. Second, they are spatially distributed, which means that there are variations between the states at one spatial

location and the other. To be more precise, the systems mostly involve the transport of a quantity in space through natural media. Third, they are often treated as stochastic. The reason is that the systems are not fully understood at larger scales, thus appearing to occur in a random manner and introducing uncertainties in the corresponding models and predictions. The source of such uncertainties can be aleatoric, which is caused by inherent randomness and is irreducible even with more information, or epistemic, which is induced by scarce data and missing knowledge or understanding of the system.

The purpose of this thesis is to fill in the missing knowledge for these types of problems. It focuses specifically on physical systems that are mathematically modeled with Partial Differential Equations (PDEs) because PDEs capture both the time dependency and spatial connectivity of a dynamic and spatially distributed system. Additionally, this thesis also concentrates on equipping the models with uncertainty quantification methods to enhance the robustness of the modeling process.

1.2 Learning from Data

The past decade has seen the rapid development of data-driven modeling approaches in many aspects. This development can be attributed to the increase in computational power and availability of data [e.g. Jordan and Mitchell, 2015, Appenzeller, 2017, Aggarwal, 2018]. Additionally, the popularity of data-driven modeling is further boosted by its powerful ability to extract information from observation data that the physics-based model lack [e.g. Gupta and Nearing, 2014]. One prominent approach is the Artificial Neural Network (ANN), which was already proven to be a universal approximator [Hornik, 1991]. In other words, ANNs are capable to extract complex relationships between variables. As a consequence, data-driven modeling in general, and ANNs in particular, could be a useful tool for discovering new knowledge from data, which is the focus of this thesis and could help reduce the epistemic uncertainty that plagues the modeling of many dynamic and spatially distributed systems.

Considerable research has grown up around the development and applications of ANNs. More complicated ANN architectures are recently proposed to tackle various types of problems. This is exemplified in the adoption of the Nonlinear Autoregressive Network with Exogenous Input (NARX) [Chen et al., 1990], Long Short-Term Memory (LSTM)

[Hochreiter and Schmidhuber, 1997], Gated Recurrent Unit (GRU) [Cho et al., 2014], Transformer [Vaswani et al., 2017], and Neural Ordinary Differential Equation (Neural ODE) [Chen et al., 2018] to solve dynamic problems as opposed to the basic Recurrent Neural Network (RNN). Furthermore, more complex variants of the Convolutional Neural Network (CNN) such as the Temporal Convolutional Network (TCN) [e.g. Bai et al., 2018] and the Convolutional LSTM (ConvLSTM) [Shi et al., 2015], have been applied to solve spatially distributed physics problems [e.g. Westermann et al., 2020, Huang et al., 2021, Li et al., 2021a]. Besides the CNN-based models, non-convolution-based models such as the Distributed Spatio-Temporal Artificial Neural Network Architecture (DISTANA) [Karlbauer et al., 2019] and Fourier Neural Operator (FNO) [Li et al., 2021b] have also been proposed to solve spatio-temporal problems. This rapid development of ANN, together with the history of successful implementations on predicting physical phenomena [e.g. Hsu et al., 1995b, Maier and Dandy, 1996, Hsu et al., 1995a, Ashena and Thonhauser, 2015], further motivate the choice of ANN as the data-driven modeling tool in this thesis.

The aforementioned approaches, however, have failed to address the interpretability issue of ANNs. ANNs are inherently black box models with inadequate explanations of what each of their elements represents [e.g. Breiman, 2001, Doshi-Velez and Kim, 2017, Miller, 2019]. Consequently, the inner workings of ANNs are not easily interpretable [e.g. Petch et al., 2022]. Additionally, ANNs require a huge amount of data to produce accurate predictions. In the physics domain, the abundance of data is not always guaranteed, due to the cost of observation or experiment [e.g. Hoffmann et al., 2019]. With limited availability of training data, ANNs often produce implausible results [e.g. Karpatne et al., 2017, 2022, Reichstein et al., 2019]. Moreover, the black-box characteristic of ANNs hinders the discovery of trustworthy new hypotheses [e.g. Murdoch et al., 2019, Carleo et al., 2019, Dybowski, 2020, Roscher et al., 2020, Oviedo et al., 2022, Freiesleben et al., 2022]. In short, humans use ANNs as a pragmatic tool, without gaining an additional understanding of the modeled systems. Because of this main drawback, this thesis focuses on administering improvements so that the ANNs develop further as a reliable tool to aid in critical decision-making processes and in extracting scientifically interpretable hypotheses on the functioning of dynamic and spatially distributed systems.

1.3 Infusing Prior Physical Knowledge

ANN and physics-based models lie on two different extremes. On one end of the spectrum, ANNs use a high amount of data to establish relationships between variables, namely the inputs and outputs, particularly when no physical knowledge exists about the modeled system. On the other end, physics-based models are based on known and proven scientific principles. A considerable amount of work recently has been conducted on fusing data-driven and physics-based modeling to design a hybrid model that profits from the strengths of both models [Karpatne et al., 2017, Karniadakis et al., 2021]. In this manner, the hybrid model should possess the flexibility and ability to learn from data, in addition to being scientifically robust and physically consistent. More importantly, it should address the interpretability problem of ANNs. This hybrid modeling framework has been proposed with different names, but in this thesis, it will be referred to as Physics Informed Neural Network (PINN) interchangeably.

The most common method of infusing physical knowledge to ANNs is to add a physically-formulated constraint into the loss function as an additional regularization term. This physics-based regularization restricts the hypothesis space so that it reduces the possibility of physically-inconsistent models being chosen as the correct model [Hastie et al., 2009]. The degree of physical information employed in such a method varies from partial to complete knowledge. Examples of partial physical knowledge include the monotonicity of the water density with regards to the depth of a lake and consistency of energy conservation over time [Jia et al., Read et al., 2019]. On the other hand, the other method assumes complete knowledge and validity of a PDE to be capable of fully describing the modeled system [e.g. Raissi et al., 2019, Tartakovsky et al.].

Both aforementioned types of knowledge inclusion differ not only in the degree of physical information introduced into the model but also in their advantages and purposes. Partial physics knowledge provides more flexibility in searching through larger hypothesis spaces because the constraints are not as restrictive as if using the full PDE. However, this type of knowledge inclusion is usually only fit for solely predicting some unknown states of the modeled system, and is not suitable for other purposes. On the contrary, the use of complete physics knowledge is more appropriate for a wider range of motivations, such as surrogate modeling and inverse problems [e.g. He et al., 2020]. More concretely, the PDE parameters can also be unknown and inferred during

the model training, because the corresponding form of the PDE is assumed to be true and representative of the modeled system. Nevertheless, the full knowledge inclusion restricts the model generalization ability to be fairly limited to the specific conditions that the model is trained on. Additionally, the strict form of PDE used to define the model also limits the expressiveness and the learning capability of the model.

Another method of infusing physical knowledge into the ANN is to design the model architecture to adhere to known physical structures [e.g. Karniadakis et al., 2021, Lavin et al., 2021]. This kind of approach ensures that certain physical properties are satisfied. The effectiveness of such methods has been exemplified in embedding invariance in an ANN to solve the Reynolds-averaged Navier–Stokes (RANS) turbulence problem [Ling et al., 2016], designing symplectic networks to ensure the conservation of Hamiltonian systems [Jin et al., 2020] and using convolutional structures to approximate spatial derivatives of PDEs in combination with data-assimilation-inspired techniques [Guen and Thome, 2020]. Such approaches, however, are mostly tailored only to a particular problem. As a result, their applicability is also severely restricted to a specific problem class.

Despite the high amount of promising work in the hybrid modeling field, there is still a huge gap that needs to be addressed before a hybrid model can be reliably used to discover new knowledge and aid in decision-making processes. More specifically, three important issues relating to this gap are identified:

1. The existing modeling frameworks either use insufficient or excessive physical knowledge. With the inclusion of physics-based regularization, the training of the model develops into a multi-objective optimization problem. A sweet spot that integrates the right amount of learning flexibility and physical restriction still needs to be found.

2. Most, if not all, of the aforementioned approaches, still need a substantial amount of training data to possess reasonable generalization. As mentioned previously, their generalization is also restricted to the specific initial and boundary conditions that they are trained on. This issue is crucial because discovered scientific knowledge should be generalizable to different conditions.

3. Existing approaches are demonstrated using synthetic datasets, and their application to real-world data, which are often noisy and sparse, is still an open question.

Furthermore, the large size of the existing models prohibits the implementation of uncertainty quantification methods [e.g. Abdar et al., 2021], which are critical in handling real-world datasets.

In this thesis, the drawbacks of the existing models are addressed. This thesis attempts to find the right combination between physics and data, and assumes a partial (soft) prior knowledge of the PDE, to enable the model to learn and discover missing knowledge from the available data. Moreover, this thesis focuses on the knowledge discovery purpose, and not on building a faster surrogate model, unlike most of the previous works. The model developed in this thesis also adopts a more general structure, that is based on well-known and commonly used numerical solvers, so that the model applies to more problems, and can generalize well to various conditions. Lastly, the purpose of this thesis is also to develop a scalable model, to facilitate uncertainty quantification methods to be performed.

1.4 Goal and Objectives

Having established the problem description and the potential possessed by the hybrid modeling approach, it can be envisioned that having a knowledge-infused data-driven model is beneficial to assist the discovery of novel scientific knowledge. This advance in the scientific domain, in turn, improves the existing prior physical knowledge and can be imposed on the ensuing hybrid model to further discover yet unidentified scientific information. What follows is an open-ended optimization cycle to continuously improve on the inductive bias administered in the hybrid model based on well-established scientific principles and its own previous discovery [Bergen et al., 2019, Lavin et al., 2021]. This framework, consequently, accelerates state-of-the-art scientific discovery and improves the model prediction and the resulting decision-making processes.

As the first step towards reaching this ultimate vision, the goal of this thesis is to design a hybrid modeling framework for spatio-temporal problems that can contribute to the reliable generation of new knowledge, while being truthful about its own uncertainty. To re-emphasize, this thesis focuses not on building a faster surrogate with restrictive prior knowledge, but rather to build a model that allows reliable learning of new scientific hypotheses from the available data. To achieve this goal, three specific

research questions are pursued, along with the corresponding objectives to answer each question:

1. **How can prior physical knowledge be adopted to design a consistent and reliable hybrid model for *dynamic* systems?**

   For this research question, a recurrent structure is chosen for the model to handle temporal dependencies of the unknown states of dynamic systems. More importantly, the recurrence should also be reflected during the training to improve long-term prediction. Moreover, additional physics-based regularization may further improve the plausibility of the model prediction. These aspects are tested in this thesis by comparing the proposed hybrid model with a conventional machine-learning model that does not experience recurrence and is not physically regularized during its training.

2. **How can prior physical knowledge be adopted to design a consistent and reliable hybrid model for *dynamic and spatially distributed* systems?**

   To answer this research question, this thesis proposes a combination of the well-established numerical structure of the Finite Volume Method (FVM) [e.g. Moukalled et al., 2016] as the physical inductive bias to handle spatio-temporal problems with the learning capability and flexibility of ANNs as the data-driven scheme. This combination is intended to provide prior structural knowledge about the translation invariance of the system's behavior and to ensure conservation properties, and it is flexible and general enough to apply to different problems. Furthermore, the proposed hybrid model should not only generalize well to different values and types of numerical boundary conditions, but should also be able to learn unknown closure/constitutive relationships from data. The designed hybrid model is then compared with existing state-of-the-art models about their prediction accuracy and consistency when applied to various equations. Additionally, the model is also tested on data generated with unseen initial or boundary conditions, and the learned closure functions are also compared to the ground truth assumed in the data generation process.

3. **How can the hybrid model learn about its *total (predictive) uncertainty***

in a computationally effective manner, so that it is appropriate for real-world applications?

To address this research question, the parameters of the hybrid model are treated as random parameters to quantify the model's systematic uncertainty, and noisy real-world data are used to represent the aleatoric uncertainty. Then, various uncertainty quantification methods are compared, such as the cheaper parameterized variational inference method [Blundell et al., 2015] and more computationally demanding Markov Chain Monte Carlo (MCMC) methods [e.g. Andrieu et al., 2003]. Because the machine learning toolbox [Paszke et al., 2019] used in this thesis provides gradient information of the model prediction, more complicated gradient-based MCMC approaches are also used to obtain better convergence of the algorithm [e.g. Sengupta et al., 2016]. The tests include comparing the convergence of various uncertainty quantification methods to determine the most appropriate and efficient algorithm for the proposed hybrid model, as well as a comparison of the learned closure function and the prediction of the proposed model with their confidence intervals, against a calibrated physical model.

1.5 Structure and Contributions

This thesis is divided into four chapters and four appendices. This first chapter lays down the problem description and motivation of this thesis, including the formulation of the research goal, objectives, and research questions. The second chapter provides explanations of dynamic and spatially distributed systems and the state-of-the-art ANN methods that are commonly used to solve them. It also summarizes uncertainty quantification methods that are employed in this thesis. To clarify, this thesis does not provide explanations on the fundamentals of ANNs. Interested readers are referred to Goodfellow et al. [2016]. The third chapter discusses the contributions of this thesis, including the summary of all related publications included in this thesis. Lastly, the fourth chapter concludes the contributions and how they relate to answering the research questions, as well as the remaining challenges and open questions to move closer to the bigger vision.

Each appendix presents a publication that supports the thesis goal. The first publication [Praditia et al., 2020] looks at improving the prediction and physical plausibility of a dynamic system, in the form of a relatively complex Thermochemical Energy Storage (TCES) system. The improvement is achieved by introducing recurrence in the model training so that the long-term dynamic prediction is reasonably accurate and stable, and by using mass and energy conservation laws to regularize the model training. This publication addresses the first research question.

The second publication [Praditia et al., 2021] and the third publication [Karlbauer et al., 2022] focus on fusing the FVM and Neural ODE frameworks to handle spatio-temporal problems. In this manner, both publications contribute to building a hybrid model that outperforms existing state-of-the-art models in terms of prediction accuracy and consistency, the generalization to different initial and boundary conditions of the modeled systems, and the ability to explicitly learn unknown closure relationships for knowledge discovery. Therefore, the second and the third publication address the second research question.

The last publication [Praditia et al., 2022] compares different uncertainty quantification methods to investigate the most computationally efficient algorithm for the proposed hybrid model and to provide a more robust prediction in the face of model and measurement uncertainties. Additionally, noise and the sparse real-world dataset are also utilized as the training data, to show the fitness of the proposed model for a real-world application and its superiority compared to a calibrated physical model that is restricted by a rigid parameterization. This fourth publication addresses the third research question.

2 Methods

This chapter provides an overview of state-of-the-art methods that are applicable for dynamic and spatially distributed systems as well as for uncertainty quantification. This chapter is arranged into three sections. Section 2.1 discusses the general form of Ordinary Differential Equations (ODEs) to define dynamic systems, and machine-learning models that are used to process them. Section 2.2 discusses the general form of Partial Differential Equations (PDEs) to define dynamic and spatially distributed problems, also called spatio-temporal problems. This section also presents existing machine-learning models that apply to solving such problems. Finally, Section 2.3 introduces uncertainty quantification methods that were built upon the Bayesian framework and are suitable for physics-based models, machine-learning models, and hybrid models.

2.1 Dynamic systems

A dynamic system is defined as a system with one or more internal states that evolve over time. The rate at which these states change characterizes the behavior of the dynamic system. Generally, this behavior is written in the form of an ODE [e.g. Arfken et al., 2013]:

$$u_{t+1} = u_t + \int_t^{t+1} f(u, t)\,\mathrm{d}t, \tag{2.1}$$

where u is the variable of interest, t is the time index, and $f(u,t) = \frac{\mathrm{d}u}{\mathrm{d}t}$ is the rate of change, which is mathematically defined as the derivative with respect to time. Note that, in this thesis, only first-order ODEs are considered.

In most physics applications, the initial condition of the system is known, and the focus is therefore to solve an initial value problem [e.g. Stroud, 1974]. To calculate

the integral in Equation 2.1, various numerical methods can be employed, starting from the simple Euler method to the more complicated Runge-Kutta method [Butcher, 2008]. The choice of the numerical integrator depends on the desired accuracy and the difficulty of the problem at hand.

To derive these numerical integration methods, a Taylor expansion of Equation 2.1 is written:

$$u_{t+1} = u_t + u'_t \Delta t + \frac{1}{2} u''_t \Delta t^2 + \mathcal{O}(\Delta t^3), \tag{2.2}$$

where Δt is the time step size, u'_t is the first derivative of u with respect to t, and u''_t is the second derivative of u also with respect to t. The Euler method approximates Equation 2.1 by truncating the Taylor expansion in Equation 2.2 right after the u'_t term. In contrast, the Runge-Kutta method calculates coefficients based on u'_t at various points in the interval $[t, t+1]$ to obtain a better approximation of the Taylor expansion with higher order. As a result, the Runge-Kutta method produces a more accurate integration of Equation 2.1 compared to the Euler method. Furthermore, the error of the numerical integration can be controlled using adaptive methods, where the step size Δt is adjusted at each evaluation step according to the truncation error.

Besides these numerical methods, ANNs also have been exploited to solve dynamic problems. The Recurrent Neural Network (RNN) [e.g. Elman, 1990] provides the simplest approach, assuming that the value of u_{t+1} depends only on its own value at an earlier point in time u_t and an external input h_t that might contain useful additional information. Therefore, the RNN model can be written as $u_{t+1} = g(u_t, h_t)$. The Nonlinear Autoregressive Network with Exogeneous Input (NARX) [Chen et al., 1990] facilitates a more complicated approach, by providing information from multiple time points in the past as well as an external input h_t. Mathematically, the NARX model is written as $u_{t+1} = g(u_t, u_{t-1}, \cdots, u_{t-d_u}, h_t)$, where h_t is the exogenous input and d_u is the maximum feedback delay of the model. More recently, the Neural Ordinary Differential Equation (Neural ODE) [Chen et al., 2018] was proposed to learn only the unknown time derivative $f(u, t)$ instead of the whole right-hand side of Equation 2.1. In the next subchapters, an overview of these ANN models is presented.

2.1.1 Recurrent Neural Networks

RNN is a series of repeated feedforward neural networks [e.g. Bebis and Georgiopoulos, 1994] that allows the processing of sequential data, including time series. At each time step, it receives the value of u from the previous time step, as well as an additional input h, to perform an evaluation of the subsequent value of u. In other words, RNN feeds back its own prediction as an input to predict the next output. Hence, it explicitly models the dependency of output at a particular time step on its past values. This flow of information within an RNN is better visualized by unrolling the network, as shown in Figure 2.1.

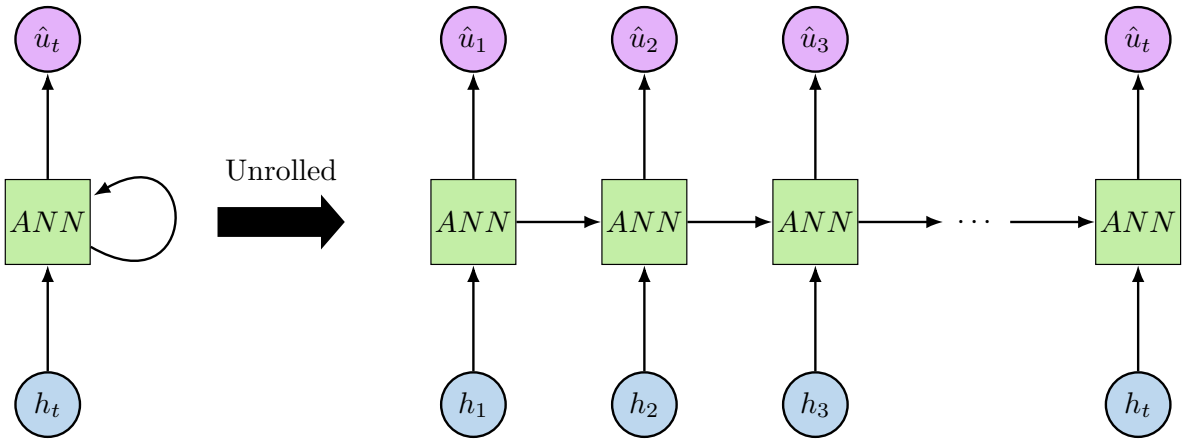


Figure 2.1: The general architecture of a Recurrent Neural Network model*

These feedforward neural networks that build the RNNs consist of multiple hidden layers that are connected through the learnable parameters w. Mathematically, these connections can be written as

$$h_i^l = a\left(\sum_{j=1}^{N_{w,l-1}} w_{i,j}^{l-1} h_{i,j}^{l-1}\right), \tag{2.3}$$

where h_i^l denotes the hidden state of node i in layer l, $N_{w,l-1}$ denotes the number of nodes in layer $l-1$, $w_{i,j}^{l-1}$ denotes the ANN parameter that connects node j from layer $l-1$ to node i in layer l, and $a(\cdot)$ is a non-linear activation function.

Because the same feedforward network is employed in an RNN for every time step, RNNs possess various advantages for time series modeling [e.g. Du and Swamy, 2013,

*Redrawn from Olah [2015].

Sharkawy, 2020]. First, the increase in sequence length scales well to the increase in computational cost to perform forward propagation, because the model size does not grow with increasing sequence length. Second, RNNs are able to process any sequence with an arbitrary length that is different from the length of the training data. Third, the time dependency of the data is captured well by the RNN structure. More importantly, the RNN structure is physically reasonable, providing an assumption that the learned physical relationship that governs the modeled system should remain the same at any given time.

To train and update the parameter values of an ANN, gradient information with respect to its parameters w is propagated backward from the output to the input [Rumelhart et al., 1986]. The gradient is calculated based on the discrepancy between the prediction $\hat{u}$ and the data u, defined in the loss function $L(\hat{u}, u)$. With RNN, the gradient backpropagation has to be performed through the unrolled network, using the method called Backpropagation Through Time (BPTT) [Werbos, 1990]. Mathematically, the process of BPTT can be written as

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{u}_t}\left(\sum_{i=1}^{t}\left(\prod_{j=i+1}^{t}\frac{\hat{u}_j}{\partial \hat{u}_{j-1}}\right)\frac{\hat{u}_i}{\partial w}\right). \tag{2.4}$$

Earlier, it was stated that one of the advantages of using RNN is that it scales well to an increasing sequence length. Unfortunately, this statement holds true only for forward propagation. During training, the summation and multiplication term in Equation 2.4 also grows with increasing sequence length, leading to reduced computational efficiency in the backpropagation process. Moreover, this growing term in the gradient calculation could also lead to either a very small value (vanishing gradient problem) or an extremely large value (exploding gradient problem) [e.g. Hochreiter et al., 2001, Bayer et al., 2009, Pascanu et al., 2013, Sarker, 2021]. Consequently, the vanishing gradient problem prevents the RNN parameters to be updated, while the exploding gradient problem imposes instability during the training. Additionally, the flow of information could also vanish during forward propagation, causing RNN to often fail in capturing long-term dependency. To alleviate this problem, state-of-the-art models have been proposed, such as the Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] and NARX [Chen et al., 1990].

2.1.2 Nonlinear Autoregressive Network with Exogeneous Input

The Nonlinear Autoregressive Network with Exogeneous Input (NARX) is an alteration of RNN that enables a feedback delay of past predictions [Chen et al., 1990]. In addition to only using u_{t-1} as an input to predict u_t, NARX also provides u at multiple preceding time steps up to a certain feedback delay d as inputs to the model, that is u_{t-2}, $\cdots$, u_{t-d}. Mathematically, the NARX model is written as

$$\hat{u}_{t+1} = g\left(u_t, u_{t-1}, u_{t-2}, \cdots, u_{t-d}, h_t\right). \tag{2.5}$$

The function $g\left(u_t, u_{t-1}, u_{t-2}, \cdots, u_{t-d}, h_t\right)$ in Equation 2.5 can also be seen as an approximation to linear multistep methods, for example, the Adams-Bashforth method [e.g. Peinado et al., 2010, Tutueva et al., 2020], which are suitable to handle problems with long-term dependencies. As a consequence, NARX is also more suitable to capture long-term dynamics compared to the RNN.

NARX has two different forms, namely the Series-Parallel (SP) and Parallel (P), as shown in Figure 2.2. The SP mode feeds the real values of u as inputs to the network, instead of the predicted values of $\hat{u}$. To put it differently, the SP mode has no recurrence and the feedback loop is open. In contrast to that, the P mode closes the feedback loop, providing the predicted values of $\hat{u}$ at the preceding time steps as the inputs to the model, behaving more like an RNN compared to the SP mode.

In previous works, NARX was commonly trained in the SP mode to achieve faster convergence and better stability during training. Only during testing is the feedback loop closed, by converting the SP mode into the P mode. However, this procedure fails to train the model well to produce accurate long-term predictions [e.g. Buitrago and Asfour, 2017, Boussaada et al., 2018]. In this thesis, the P mode is implemented both during training and testing, because training in P mode minimizes the effect of error accumulation over time, in addition to learning the long-term dynamics better.

2.1.3 Neural Ordinary Differential Equations

Up until recently, various ANN models have been used to solve ODEs only in the discrete temporal domain. The Residual Network (ResNet) [He et al., 2016] is one

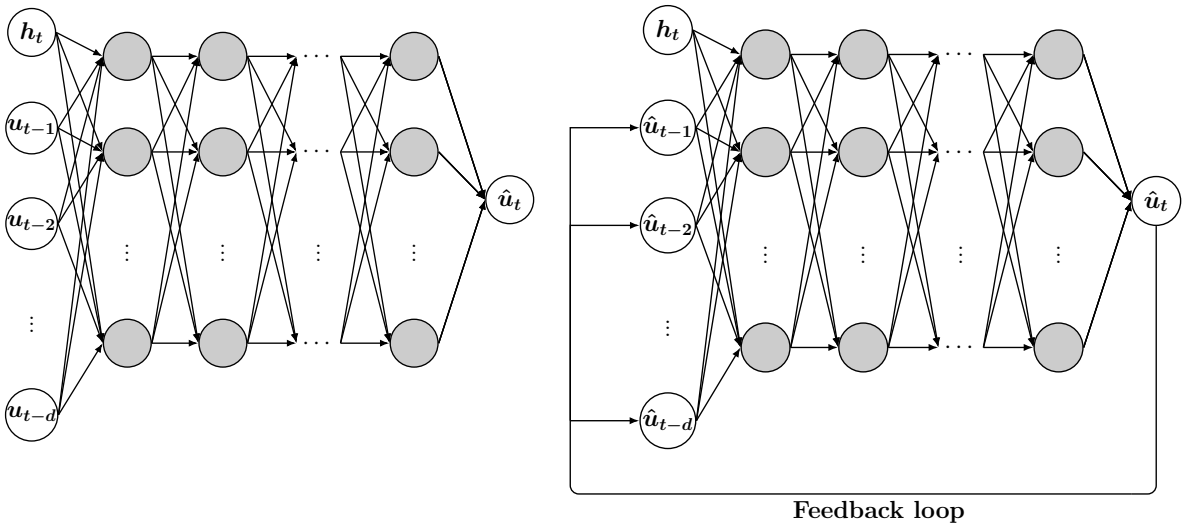


Figure 2.2: Two different modes of NARX: (a) Series-Parallel and (b) Parallel

example, which imitates the Euler discretization of an ODE by introducing the skip connection in its structure [Lu et al., 2018]. It is only since the Neural Ordinary Differential Equation (Neural ODE) method was proposed [Chen et al., 2018] that the study of learning continuous dynamics has developed into a popular topic. The Neural ODE learns the time derivative $f(u, t)$ as a representation of the continuous system dynamics, and integrates these learned dynamics using an ODE solver of choice.

Since many ODE solvers are black boxes, backpropagation often can not be performed through the solvers. As a solution, the gradient calculation is executed using the adjoint sensitivity method [e.g. Pontryagin et al., 1987]. For this purpose, a new adjoint state a_t is introduced, which is defined as the derivative of the loss function with regards to the unknown variable: $a_t = \partial L / \partial u_t$. The dynamics of the adjoint state are defined as

$$\frac{\partial a_t}{\partial t} = -a_t \frac{\partial f(u_t, t, w)}{\partial u_t}, \tag{2.6}$$

where $f(u_t, t, w)$ is the time derivative of the system states $\partial u_t / \partial t$, learned by an ANN that is parameterized by w.

With the use of the adjoint state, the gradient about the model parameters can finally be defined with

$$\frac{\partial L}{\partial w} = -\int_{t_1}^{t_0} a_t \frac{\partial f(u_t, t, w)}{\partial w} \, \mathrm{d}t, \tag{2.7}$$

where t_0 is the initial time step and t_1 is the final time step. Next, both the adjoint state and the gradient are solved backward in time from t_1 to t_0, by using an ODE solver to integrate Equation 2.6 and Equation 2.7 with the initial conditions $a_1 = \partial L/\partial u_1$ and $\partial L/\partial w = 0$, respectively. For a complete derivation of the adjoint method, interested readers are referred to Chen et al. [2018].

The Neural ODE model has considerable benefits in comparison to other ANN models that directly learn the discrete representation $f(u,t)\Delta t$. Learning the time derivative $f(u,t)$ in the continuous domain allows the incorporation of data with an irregular time step size, whereas learning $f(u,t)\Delta t$ together implies that the time step size has to be constant throughout the simulation. Accordingly, Neural ODE is more appropriate to operate on real-world observation data, which are often measured at arbitrary intervals. Furthermore, the continuous definition of the time derivative facilitates the use of higher-order integration methods for better accuracy, computational efficiency, and numerical stability. Because of these desirable properties, Neural ODE is employed in this thesis.

2.2 Spatially distributed systems

In a dynamic and spatially distributed system (which is also referred to as a spatio-temporal problem), the internal states vary not only in time but also in space. Accordingly, the spatial derivative is also necessary to define a dynamic and spatially distributed system, in addition to the time derivative. In order to impose relationships between derivatives with regard to multiple variables, a Partial Differential Equation (PDE) is required [e.g. Benkirane and Touzani, 2002].

This thesis focuses on PDEs that are first order in time and second order in space, more specifically the advection-diffusion-reaction (ADR) equations [e.g. Friedlander et al., 2002] because they are ubiquitous and useful to describe numerous physical phenomena. Prominent applications of the ADR equations have been shown in the modeling of fluid dynamics [e.g. Ferziger and Peric, 2012], heat transfer [e.g. Hongren, 1994], and biological pattern formations [Turing, 1952].

The ADR equations are generally derived from conservation laws and are mathematically written as

$$S\frac{\partial u}{\partial t} = \nabla \cdot (D\nabla u) - \nabla \cdot (vu) + q, \tag{2.8}$$

where u is the variable of interest, t is the time, S is the storage coefficient, D is the diffusion coefficient (or tensor), v is the advection velocity, and q is the source/sink term. To explain the advection-diffusion process, the right-hand side of Equation 2.8 can be broken down into different parts. The first part, $\nabla \cdot (D\nabla u)$, describes the diffusion process, that is the movement of a substance from a region with a high value of u to the surrounding regions with lower values of u. The second part, $\nabla \cdot (vu)$, describes the advection process, that is the transport of a substance from one region to another by a bulk motion with a certain velocity v. While the diffusion process is defined by a second-order spatial derivative, the advection process is defined by a first-order spatial derivative. The third part, q, describes the generation or elimination of the substance, for example by a chemical reaction. The combination of all three terms balances into the storage term $S\,\partial u/\partial t$, which defines the change of the value u over time. The storage coefficient S can for example be unity in bulk fluids, equal to porosity in porous media, or describe sorption mechanisms. All coefficients S, D, v, and q can either be constants or depend on u. Accordingly, Equation 2.8 can also either be a linear or non-linear PDE.

The ADR equations are commonly solved with various numerical methods, namely the Finite Difference Method (FDM) [e.g. Morton and Mayers, 1994], the Finite Volume Method (FVM) [e.g. Moukalled et al., 2016], and the Finite Element Method (FEM) [e.g. Logan, 1992]. These methods exploit different strategies to approximate the spatial derivatives, such as the Taylor expansion (FDM), Gauss' divergence theorem (FVM), and combinations of basis functions (FEM). Among all these methods, FVM is the method that is derived based on fluxes as physical properties, and therefore it ensures the conservation and physical consistency of the solution. Additionally, the FVM is relatively simple to implement compared to the FEM. Because of these characteristics, FVM is employed in this thesis so that the integration with ANNs is more straightforward, and the resulting hybrid model is physically interpretable. A short overview of the FVM discretization is provided in the next section.

Besides physics-based models, numerous deep-learning models have also been proposed to solve PDEs. Many of such models use a convolutional structure to represent the

involved spatial derivatives, such as the Temporal Convolutional Network (TCN) [Bai et al., 2018], Convolutional LSTM (ConvLSTM) [Shi et al., 2015], and PhyDNet [Guen and Thome, 2020]. Another model such as the Distributed Spatio-Temporal Artificial Neural Network Architecture (DISTANA) [Karlbauer et al., 2019] employs a kernel to quantify the movement of a substance from one place to another. Another group of models represents the unknown variable u as an explicit function of its location in the spatial domain. This group includes models such as the Physics-Informed Neural Network (PINN) [Raissi et al., 2019] and the Fourier Neural Operator (FNO) [Li et al., 2021b]. These models are implemented to benchmark the performance of the proposed model in this thesis, and therefore, brief overviews of these models are also presented in the upcoming sections.

2.2.1 Finite Volume Method

The Finite Volume Method solves PDEs by converting the continuous spatial domain into discrete control volumes. In each control volume, a volume integral of the PDE is evaluated, resulting in

$$\int_{\nu_i} S\frac{\partial u}{\partial t}\,\mathrm{d}\nu = \int_{\nu_i} \nabla\cdot(D\nabla u)\,\mathrm{d}\nu - \int_{\nu_i} \nabla\cdot(vu)\,\mathrm{d}\nu + \int_{\nu_i} q\,\mathrm{d}\nu, \tag{2.9}$$

where ν_i denotes the control volume with the index i. Volume integrals of the divergence both in the diffusion and advection term are transformed into surface integrals by employing the Gauss' divergence theorem [e.g. Arfken et al., 2013].

The surface integral of the diffusion term is calculated over the enclosing control volume surfaces, leading to the following equation:

$$\int_{\nu_i} \nabla\cdot(D\nabla u)\,\mathrm{d}\nu = \oint_{\Gamma} (D\nabla u)\cdot\hat{n}\,\mathrm{d}\Gamma, \tag{2.10}$$

where Γ is the control volume surface and $\hat{n}$ is the normal vector pointing out of Γ. By using a central difference scheme to approximate the gradient ∇u, and assuming a one-dimensional spatial domain for conciseness, this surface integral is discretized into

$$\oint_{\Gamma} (D\nabla u)\cdot\hat{n}\,d\Gamma \approx A_{i-1}D_i\frac{u_{i-1}-u_i}{\Delta x} - A_{i+1}D_i\frac{u_i-u_{i+1}}{\Delta x}, \tag{2.11}$$

where A_{i-1} and A_{i+1} are the surface areas of control volume i on the left and right, respectively, and Δx is the length of the control volume.

A similar surface integral is applied for the advection term, leading to the following equation:

$$\int_{\nu_i} \nabla \cdot (vu)\,\mathrm{d}\nu = \oint_{\Gamma} (vu) \cdot \hat{n}\,\mathrm{d}\Gamma. \tag{2.12}$$

However, the implementation of a central difference scheme in the advection term could lead to numerical instability issues. To alleviate this, the upwind difference scheme is often preferred to the central difference scheme [e.g. Versteeg and Malalasekera, 1995], resulting in the discrete equation:

$$\oint_{\Gamma} (vu) \cdot \hat{n}\,\mathrm{d}\Gamma \approx A_{i-1} \max(v_i, 0)\,(u_i - u_{i-1}) + A_{i+1} \min(v_i, 0)\,(u_{i+1} - u_i). \tag{2.13}$$

Here, the upwind direction is determined by the direction of the velocity v_i. If $v_i > 0$, then u_{i-1} is the upwind side (the flow direction is from left to right), and therefore, u_{i+1} is ignored. As a result, Equation 2.13 reduces to $A_{i-1}v_i(u_i - u_{i-1})$. The reverse condition is valid for $v_i < 0$, where u_{i+1} becomes the upwind side, and the equation reduces to $A_{i+1}v_i(u_{i+1} - u_i)$.

By substituting Equation 2.11 and Equation 2.13 into Equation 2.8, as well as integrating the storage and source/sink term over the control volume, a spatially discrete form of the advection-diffusion equation can be written as the following:

$$\begin{aligned} S_i \frac{\partial u_i}{\partial t} \nu_i =& A_{i-1} D_i \frac{u_{i-1} - u_i}{\Delta x} - A_{i+1} D_i \frac{u_i - u_{i+1}}{\Delta x} \\ & - \left(A_{i-1} \max(v_i, 0)\,(u_i - u_{i-1}) + A_{i+1} \min(v_i, 0)\,(u_{i+1} - u_i)\right) + q_i \nu_i. \end{aligned} \tag{2.14}$$

The resulting discrete equation represents a system of coupled ODEs, defined for each control volume $i = 1, \ldots, N_x$, where N_x is the number of control volumes. These ODEs are then solved with any ODE solver of choice, as discussed in Subchapter 2.1. Since numerical integration of this ODE system is only conditionally stable [e.g. Courant et al., 1967, Isaacson and Keller, 1994], an ODE solver with time step adaptivity is preferable to provide better numerical stability, in addition to being computationally more efficient.

2.2.2 Temporal Convolutional Networks

The Temporal Convolutional Network (TCN) [Bai et al., 2018] model processes both spatial and temporal dependency of the data by exploiting a convolutional neural network structure [e.q. Lecun et al., 1998, LeCun et al., 1999]. A convolution operation in a neural network is illustrated in Figure 2.3 on the left. In the illustration, the convolution operation is performed by shifting a 3×3 filter from the top left to the bottom right of a two-dimensional input. For each portion of the input (3×3), its dot product with the filter is calculated to produce a value of a cell in the output. By training the model, the values in the filters are updated to produce the desirable results. In the illustration, the input is padded with the values of zero to keep the output size the same as the original input size. Mathematically, a two-dimensional convolution operation can be written as

$$\hat{u}_{ij} = \sum_{a=1}^{k} \sum_{b=1}^{k} w_{ab} h_{(i+a)(j+b)}, \tag{2.15}$$

where $\hat{u}_{ij}$ is the output at row i and column j, k is the size of the filter, w_{ab} are the parameters of the filter, and h is the input. Furthermore, it is worth noting that a convolutional filter can be considered as an analog to the stencil used in numerical methods such as FVM [Rackauckas et al., 2020].

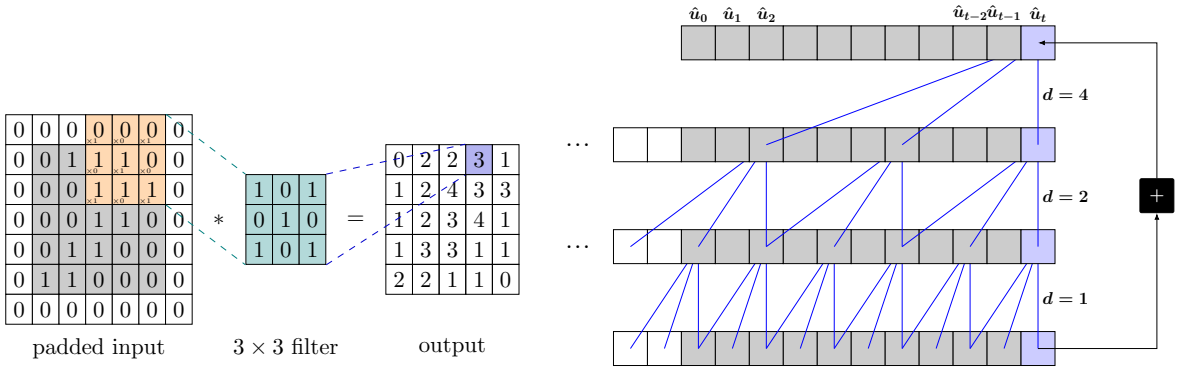


Figure 2.3: Illustration of the (a) spatial[†] and (b) temporal convolutional operation[‡]

In the temporal domain, TCN accommodates additional features that characterize its

[†]Redrawn from Veličković [2016].
[‡]Redrawn from Bai et al. [2018].

convolution operation uniquely. First, TCN capitalizes on dilated convolutions, which skip a certain amount of pixels to enlarge the temporal receptive (input) field. As illustrated in Figure 2.3 on the right, the dilation grows exponentially with increasing model depth. This allows the predicted $\hat{u}_t$ to be directly influenced by inputs coming from further in the past, even from the beginning of the sequence. Consequently, TCN processes longer sequences of data more effectively. Second, to preserve the logical ordering of temporal data, it is important that the prediction is calculated based on only past values, and not future values. The temporal convolution in TCN is therefore written as

$$\hat{u}_t = \sum_{i=0}^{k-1} w_i h_{t-di}, \tag{2.16}$$

where $\hat{u}_t$ is the prediction at time step t, k is the filter size, h is the input, and d is the dilation factor.

All in all, TCN provides a structured way of processing both spatial and temporal correlation of the data, with the possibility of having a large receptive field to facilitate long-term dependency more effectively. However, TCN also possesses a few drawbacks. Because it uses a convolutional structure to process the spatial dependency with zero or mirror padding at the margins, it is not suitable to handle more complex types of boundary conditions such as Neumann and Cauchy, which are defined by spatial derivatives and need specific treatment at domain boundaries. Additionally, the generalization ability of TCN also suffers when it predicts sequences of different lengths compared to the training data [Bai et al., 2018].

2.2.3 Convolutional Long Short-Term Memory

Similar to TCN, the Convolutional Long Short-Term Memory (ConvLSTM) model [Shi et al., 2015] exploits a convolutional neural network structure to process the spatial dependency between the data points. The temporal correlation, however, is represented using a Long Short-Term Memory (LSTM) model [Hochreiter and Schmidhuber, 1997]. Therefore, ConvLSTM differs from the original LSTM model in the sense that ConvLSTM facilitates convolution in its internal operations.

As illustrated in Figure 2.4, ConvLSTM consists of multiple gating mechanisms. Math-

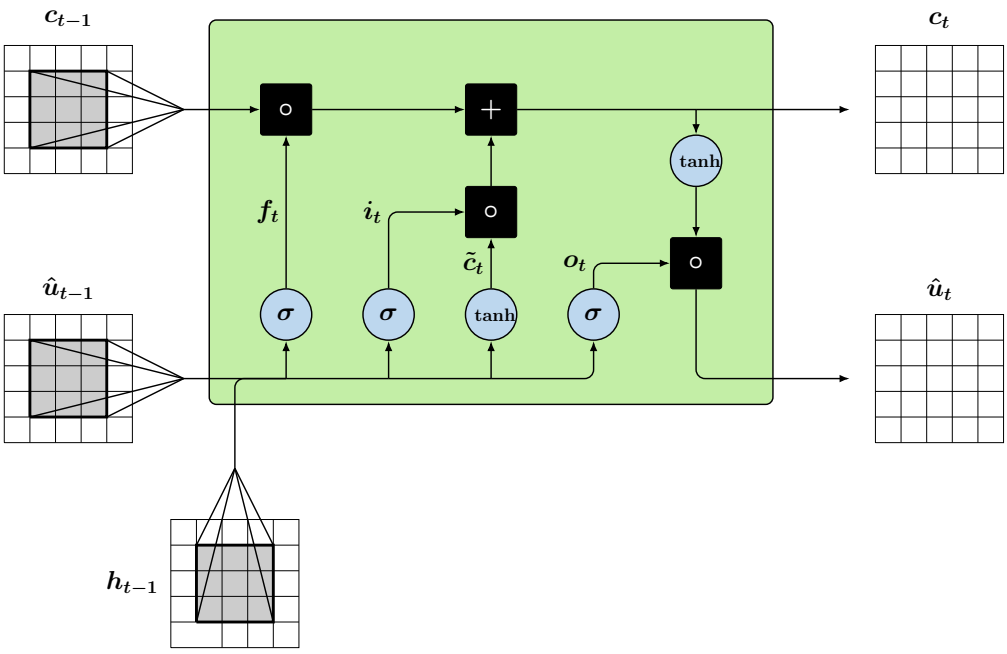


Figure 2.4: Illustration of a ConvLSTM model[§]

ematically, the internal operations of ConvLSTM are written as the following:

$$f_t = \sigma(\hat{u}_{t-1} * w_{uf} + h_{t-1} * w_{hf}), \tag{2.17}$$

$$i_t = \sigma(\hat{u}_{t-1} * w_{ui} + h_{t-1} * w_{hi}), \tag{2.18}$$

$$\tilde{c}_t = \tanh(\hat{u}_{t-1} * w_{uc} + h_{t-1} * w_{hc}), \tag{2.19}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t, \tag{2.20}$$

$$o_t = \sigma(\hat{u}_{t-1} * w_{uo} + h_{t-1} * w_{ho}), \tag{2.21}$$

$$\hat{u}_t = o_t \circ \tanh(c_t), \tag{2.22}$$

where f_t is the forget gate, i_t is the input gate, $\tilde{c}_t$ is the new processed information, c_t is the cell state, o_t is the output gate, h_t is the input, and $\hat{u}_t$ is the prediction at time t. Additionally, σ represents the sigmoid function, tanh represents the hyperbolic tangent function, $\hat{u} * w$ represents the convolutional operation between $\hat{u}$ and the model parameters w, and $\circ$ represents the Hadamard product.

First, the prediction from the previous time steps $\hat{u}_{t-1}$ and the input h_{t-1} go through

[§]Redrawn from Olah [2015].

the convolution operations in the forget gate f_t (Equation 2.17). The value of f_t is constrained between 0 and 1 due to the sigmoid function. This means that, if $f_t = 0$, past information is completely discarded. If $f_t = 1$, past information is completely kept. Next, similar operations are performed in the input gate i_t (Equation 2.18). The input gate is also constrained by the sigmoid function. Therefore, if $i_t = 0$, then new information is unused, and vice versa. To update the cell state, new information $\tilde{c}_t$ is merged with past information c_{t-1}, weighted by each importance that is learned by the input gate and forget gate, respectively (Equation 2.20). Lastly, the updated cell state c_t goes through the output gate o_t, which determines the value for the new prediction $\hat{u}_t$ (Equation 2.21 and Equation 2.22).

The gating mechanisms in ConvLSTM facilitate better passage of information for a longer time sequence because the cell state that represents the model's memory experiences only minimal disruptions. Nonetheless, ConvLSTM adopts a convolutional structure to process the information exchange in the spatial domain just like TCN. For this reason, ConvLSTM is also not suitable for handling complex boundary condition types. Moreover, ConvLSTM is also more prone to overfitting due to its complex architecture and number of parameters, therefore impacting its generalization ability [e.g. Chung et al., 2014].

2.2.4 PhyDNet

The PhyDNet model [Guen and Thome, 2020] was developed with the purpose of learning both the physical representation of the modeled system, as well as the residual unknown dynamics from observation data. This task is achieved by separating the model into two branches: a physical cell (PhyCell), which is responsible for the physics learning tasks, and a ConvLSTM cell, which is responsible for the residual learning tasks.

PhyCell is designed to update a latent state h_{t-1} using a physical predictor, corrected with a learned Kalman gain. This process resembles the Kalman filtering method for data assimilation [Kalman, 1960], with the difference that the Kalman gain in PhyCell is learned during the model training, and not calculated according to the original Kalman filter algorithm. The flow of information within PhyCell is illustrated in Figure 2.5.

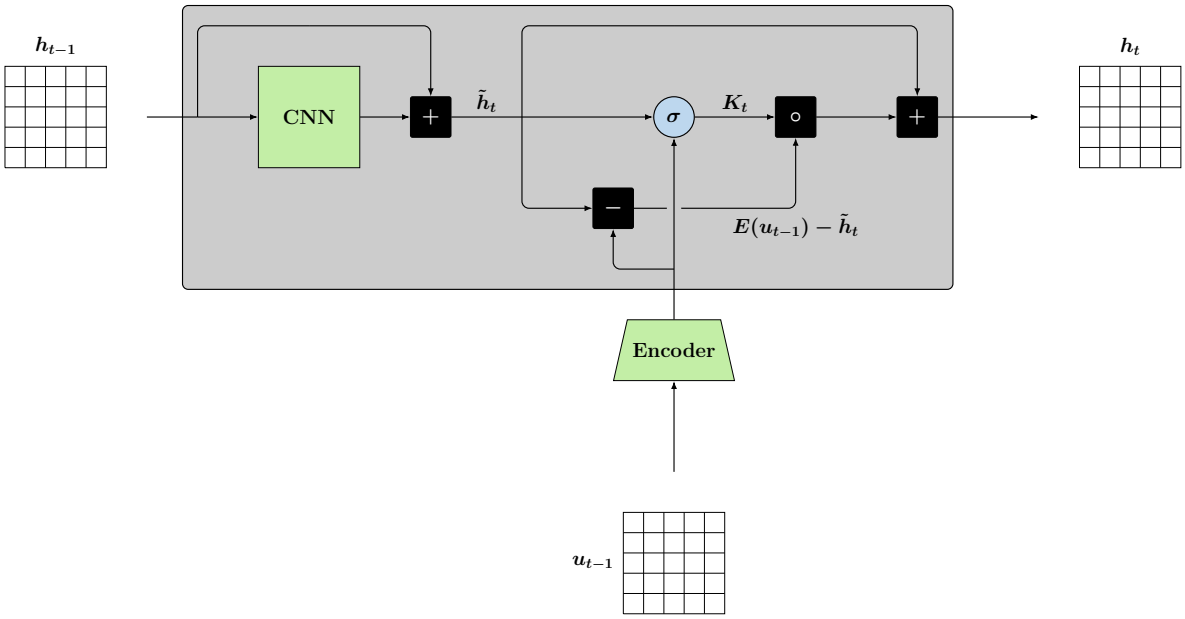


Figure 2.5: Illustration of a physical cell (PhyCell) in the PhyDNet model[¶]

The physical predictor part of PhyCell is a convolutional operation that represents the spatial derivatives of a PDE, equipped with a residual block to imitate the Euler temporal discretization method. This physical predictor takes a latent state from the previous time step h_{t-1} as an input and outputs its updated state $\tilde{h}_t$. PhyCell then encodes the value of u_{t-1} into the latent space using an encoder $E(u_{t-1})$. Together with the updated latent state $\hat{h}_t$, the Kalman gain K_t can be learned with the following equation:

$$K_t = \sigma(\tilde{h}_t * w_h + E(u_t) * w_u), \tag{2.23}$$

where w is the model parameter and $*$ is a convolution operator.

The learned Kalman gain is used to assign importance for each the physical prediction $\tilde{h}_t$ and the encoded observation $E(u_t)$. The corrected latent state can then be calculated using the Kalman gain:

$$h_t = \tilde{h}_t + K_t \circ (E(u_t) - \tilde{h}_t). \tag{2.24}$$

Note that, as an implication of the sigmoid function, the values of the learned Kalman gain are bound between 0 and 1. Accordingly, if $K_t = 0$, the input u_t is completely ignored. On the other hand, if $K_t = 1$, the updated latent state $\tilde{h}_t$ is ignored instead. Finally, the new latent state h_t predicted by the PhyCell is merged with the

[¶]Redrawn from Guen and Thome [2020].

learned residual r_t from the ConvLSTM branch. The summation of the outputs of both branches is then decoded from the latent space back into the physical space using a decoder D, resulting in the prediction for the next time step $\hat{u}_t = D(h_t + r_t)$.

Overall, PhyDNet provides a very structured way of disentangling the learning of the physical process with the learning of unknown residuals. Nevertheless, the physical processes are learned in the latent space, and therefore, they are difficult to interpret in the physical domain. It is also noteworthy that the size of the ConvLSTM branch is often significantly larger compared to the PhyCell, putting more emphasis on residual learning, as opposed to physics learning.

2.2.5 Distributed Spatio-Temporal Artificial Neural Networks

The Distributed Spatio-Temporal Artificial Neural Network (DISTANA) model [Karlbauer et al., 2019] introduces two unique kernels, namely the transition kernel and the prediction kernel, to process spatial information routing and temporal dynamics, respectively. More specifically, the transition kernels collect and process information h^{pk}_{t-1} from neighboring cells, and then transmit the processed information h^{tk}_{t-1} to their adjacent cells. The prediction kernels then receive this processed information from the transition kernels along with the prediction from the previous time step $\hat{u}_{t-1}$, and transform them to predict the unknown variable at the subsequent time step $\hat{u}_t$. An illustration of the DISTANA model is shown in Figure 2.6.

Both the prediction kernel and transition kernel are generally constructed with a similar structure. This structure consists of an encoder to transform the input into latent space, followed by an LSTM cell to facilitate recurrence and long-term memory of past information, and finally a decoder to transform the processed information back into its original space. The difference between the prediction and transition kernel lies in the type of input data that they receive. Whereas transition kernels receive only lateral information from adjacent cells, prediction kernels also receive dynamic information from their previous prediction.

DISTANA provides an elegant approach to handle spatio-temporal correlation of data, similar to a Message Passing Neural Network (MPNN) [Gilmer et al., 2017]. More concretely, the transition kernels aggregate information from neighboring cells, analogous

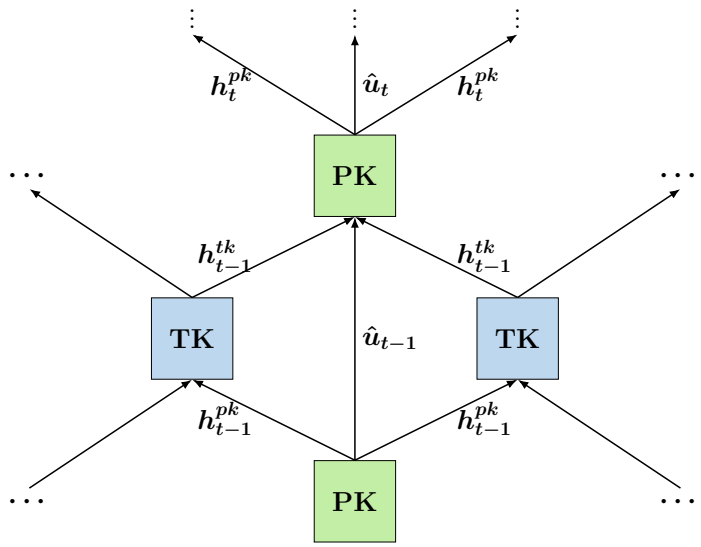


Figure 2.6: Illustration of the prediction and transition kernels of a DISTANA model[‖]

to the message-passing function on the graph edges in the MPNN. Furthermore, the prediction kernels update the dynamic prediction, similar to the node update function in the MPNN. Unfortunately, the lateral information processing performed by the transition kernels in DISTANA introduces communication latency. As a consequence, two simulation steps are required to propagate lateral information to an adjacent cell. This also means that the receptive field of DISTANA is smaller than in the other models. Additionally, DISTANA has the potential to overfit when it is trained with insufficient data, due to the lack of physical knowledge incorporated in the model.

2.2.6 Physics-Informed Neural Networks

The Physics-Informed Neural Network (PINN) [Raissi et al., 2019] is a recent major breakthrough in the field of scientific machine learning. It introduces an exquisite method to train the ANN to fit the PDE by capitalizing on the automatic differentiation method. As illustrated in Figure 2.7, PINN models the unknown variable as an explicit function of its location in space x and time t, constrained through a specific PDE form.

In vanilla ANNs, the optimization algorithm only requires gradient information with regard to the model parameters. With PINN, the backpropagation is executed ad-

[‖]Redrawn from Karlbauer et al. [2019].

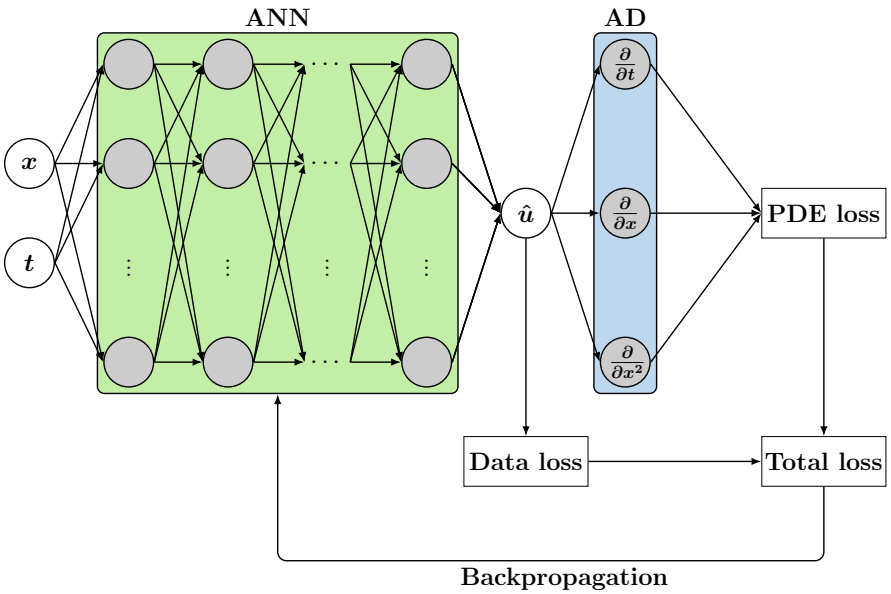


Figure 2.7: Illustration of the PINN model**

ditionally to calculate the derivative of the prediction with regards to the input x and t. This derivative information is then used to calculate the physics loss function that is defined to satisfy the governing PDE. Taking a one-dimensional, homogeneous, advection-diffusion equation as an example, the form of the physics loss function is written as

$$L_{phy} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left(\frac{\partial u}{\partial t} + v \frac{\partial u}{\partial x} - D \frac{\partial^2 u}{\partial x^2} - q \right)^2 . \tag{2.25}$$

Because Equation 2.25 can be calculated without any observation data, the training procedure of PINN is data-efficient. The observation data, however, are still useful to enforce the data-driven feature of PINN, in addition to providing additional information such as the initial and boundary conditions. The observation data define the data-driven loss function as

$$L_{data} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left(\hat{u}_i - u_i \right)^2 , \tag{2.26}$$

where $\hat{u}$ and u are the prediction and the observation, respectively. The overall loss function is then defined as the combination of both the physics and data-driven loss, that is $L = L_{phy} + L_{data}$.

**Redrawn from Lavin et al. [2021].

Despite its robust theoretical background and promising results, PINN still possesses significant flaws. Because it models the unknown variable as an explicit function of x and t, its validity no longer holds with different initial or boundary conditions. As a direct consequence, a trained PINN model can not be generalized to different modeling scenarios. Moreover, the accuracy of PINN's prediction often deteriorates substantially when it is trained on more complex PDEs [e.g. Markidis, 2021, Chiu et al., 2022].

2.2.7 Fourier Neural Operators

The Fourier Neural Operator (FNO) model [Li et al., 2021b] differs from the other deep learning models in its approach to solving PDEs. Most models are trained to approximate functions that map the input to the output in the physical space. FNO, on the other hand, approximates these functions in the Fourier space. The idea of FNO originates from the fact that differential equations in the physical space can be solved using multiplications in the Fourier domain [e.g. Bracewell and Bracewell, 1986]. Consequently, FNO devises an easier learning problem.

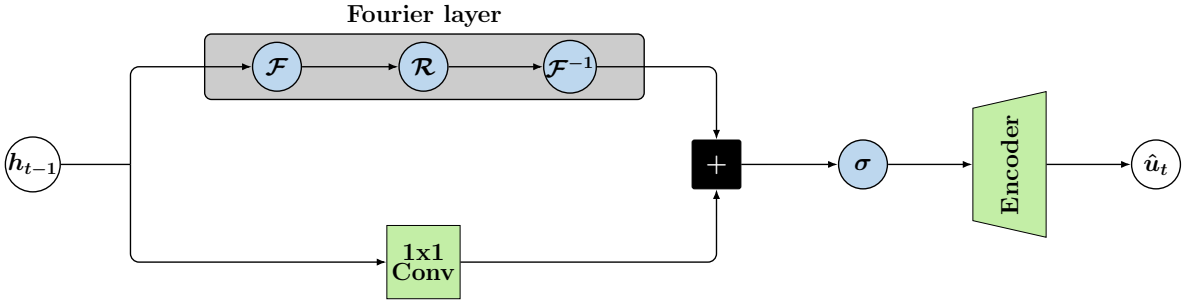


Figure 2.8: Illustration of the Fourier layer in the FNO model[††]

As inputs, the FNO receives not only the prediction from the previous time step $\hat{u}_{t-1}$, but also information on the spatial location x. These inputs are then encoded into a latent variable h_{t-1}. The encoder is succeeded by a Fourier layer, as illustrated in Figure 2.8. The latent variable enters the Fourier layer, and it is first transformed into the Fourier domain $\mathcal{F}$. Then, a learnable linear transform $\mathcal{R}$ is directly performed on the Fourier transform. The unique thing about this linear transform is that it also filters out the higher frequency modes to avoid overfitting. Finally, inverse Fourier

[††]Redrawn from Li et al. [2021b].

transforms $\mathcal{F}^{-1}$ are performed to convert back to the original space. The operations in the Fourier layer are therefore written as

$$\mathcal{K}_t = \mathcal{F}^{-1}\left(\mathcal{R}\big(\mathcal{F}(h_{t-1})\big)\right). \tag{2.27}$$

In addition to the Fourier layer, a 1×1 convolutional operation, that is a convolution layer with a filter size of 1×1, is also applied on the latent variable h_{t-1} to act as a bias, as well as to improve the flow of information in a recurrent structure. After adding the convolved feature with the output of the Fourier layer, the latent variable goes through a non-linear activation layer, and then it is encoded back into the physical space to predict $\hat{u}_t$. Mathematically, the whole process is written as

$$\hat{u}_t = E\left(\sigma(h_{t-1} * w + \mathcal{K}_t)\right), \tag{2.28}$$

where w is the filter parameter of the 1×1 convolution.

Because FNO is a mesh-independent model like PINN, it can be used to interpolate the prediction in the spatial domain, to increase the fidelity of the solution. Additionally, FNO is very stable for long-term prediction, outperforming other deep learning models. Unfortunately, FNO requires a relatively high amount of training data to achieve reasonable accuracy. Furthermore, FNO does not generalize well to different PDE parameter values [Li et al., 2021c].

2.3 Uncertainty Quantification

Observation data of an occurring natural phenomenon are often noisy and only available in a limited amount. Combined with the inherent black-box characteristic of ANNs, they could cause the model to overfit, and in turn, obstruct the model from generalizing to unseen events. Furthermore, this also causes ANNs to be generally overconfident about their predictions despite them being inaccurate. This is highly problematic, especially when a model is intended to aid in a critical decision-making process [e.g. Goan and Fookes, 2020, Jospin et al., 2022]. Consequently, it is essential to build a model that is able to systematically provide a confidence measure of its own prediction

or learned physical relationship, in order to inform about the uncertainties and possible consequences that are associated with a specific action.

A robust way to build such a model has been proposed in the form of a Bayesian Neural Network (BNN) [Mackay, 1995], which exploits the Bayesian inference method to teach the model to learn about uncertainties from the available data. BNN treats its parameters as random variables with learnable distributions instead of point estimates, as illustrated in Figure 2.9. With Bayes' theorem, a prior distribution $p(w)$ of the model parameters w is first defined. This prior distribution represents an initial idea of how the model should be, and it can take many forms, such as Gaussian and Laplacian (to enforce the sparsity of parameters). Moreover, physical knowledge of the modeled system can also be included in the prior distribution definition, for example by defining a physics violation error distribution that is centered around zero.

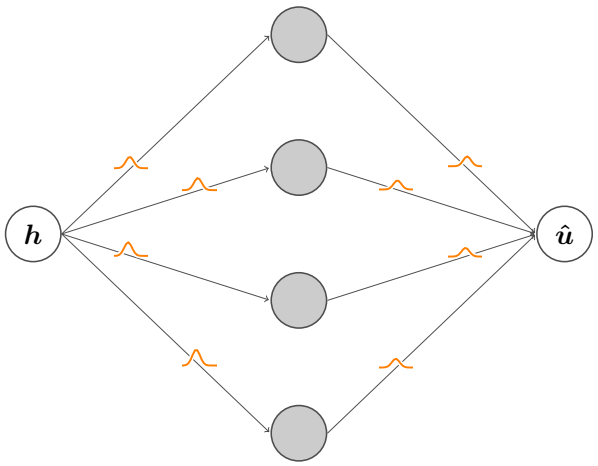


Figure 2.9: Illustration of a Bayesian Neural Network model[‡‡]

Next, the available training data D are used to update the parameter distribution through the likelihood $p(D|w)$. The likelihood quantifies the probability of observing the data given a set of model parameters. A higher value of likelihood signifies a better fit of the model parameters. Finally, the distribution is normalized by the model evidence $p(D)$, which is a marginal distribution over the whole parameter space. Since the parameter space grows exponentially with increasing dimension, the model evidence is often intractable [e.g. Schöniger et al., 2014]. Therefore, it is often enough to define the posterior distribution $p(w|D)$ of the weight given the data to be only proportional

[‡‡]Redrawn from Riebesell [2021].

to the prior distribution multiplied by the likelihood. Mathematically, this is written as

$$p(w|D) \propto p(D|w)p(w). \tag{2.29}$$

Many algorithms have been proposed to perform Bayesian inference. In the BNN setting, the most prominent algorithms include the variational inference using the Bayes by Backprop method [Blundell et al., 2015], and various Markov Chain Monte Carlo (MCMC) methods [e.g. Richey, 2010]. The Bayes by Backprop algorithm adopts a Gaussian variational posterior to define the distribution of the model parameters, which is learned through backpropagation. The MCMC algorithm, on the other hand, learns the exact posterior distribution through sampling via a Markov chain. In this thesis, three different MCMC sampling strategies are considered: Metropolis-Hastings [e.g. Chib and Greenberg, 1995], the Metropolis-Adjusted Langevin Algorithm (MALA) [e.g. Dwivedi et al., 2019], and the Barker proposal [Livingstone and Zanella, 2022]. Brief overviews of both the Bayes by Backprop and MCMC methods are provided in the following sections.

2.3.1 Bayes by Backprop

Bayes by Backprop [Blundell et al., 2015] is an algorithm that facilitates variational inference methods on BNNs. Instead of performing an exact Bayesian inference, the variational inference method simplifies the process by approximating the posterior distribution of the model parameters w with a variational distribution $q(w|\theta)$. In the Bayes by Backprop method, the variational distribution is chosen to be Gaussian, parameterized by a set of distributional parameters $\theta = (\mu, \rho)$, where μ defines the mean value and ρ defines the standard deviation.

The goal of the Bayes by Backprop method is to optimize μ and ρ through backpropagation. However, backpropagation can not be performed through a random variable. To mitigate this issue, a local reparameterization trick [Kingma and Welling, 2014] is employed through a deterministic transformation

$$w = \mu + \log\left(1 + \exp(\rho)\right) \cdot \epsilon, \tag{2.30}$$

where ϵ is a normal random noise with $\mathcal{N}(0, I)$. In this manner, ϵ is the only source of stochasticity, enforcing differentiability through the model parameters w and their distributional parameters μ and ρ.

To fit the variational distribution $q(w|\theta)$ to approximate the posterior distribution $p(w|D)$, the Kullback-Leibler (KL) divergence [e.g. Joyce, 2011] (the measure of distance between two probability distributions) should be minimized. The simplified loss function can then be derived from this KL divergence formulation as

$$\begin{aligned} L &= D_{KL}[q(w|\theta)||p(w)] - \mathbb{E}_{q((w|\theta)}[\log p(D|w)] \\ &\approx \sum_{i}^{N} \log q(w^{(i)}|\theta) - \log p(w^{(i)}) - \log p(D|w^{(i)}), \end{aligned} \tag{2.31}$$

where N denotes the number of parameter samples drawn from $q(w|\theta)$. The training algorithm of Bayes by Backprop is summarized in Algorithm 1. Note that the parameter update step can be performed not only using the gradient descent algorithm, but also with other stochastic optimization algorithms such as ADAM [Kingma and Ba, 2015].

Algorithm 1 Bayes by Backprop training algorithm

Set $\theta = \theta_0$
for $i = 1$ to N_{iter} **do**
  Draw $\epsilon \sim \mathcal{N}(0, I)$
  Set the model parameters w according to Equation 2.30
  Calculate the loss function L according to Equation 2.31
  Backpropagate and calculate the gradient $\frac{\partial L}{\partial \theta}$
  Update the distributional parameters $\theta \leftarrow \theta - \alpha \frac{\partial L}{\partial \theta}$
end for

By approximating the posterior distribution in this manner, Bayes by Backprop provides a relatively scalable fashion to train BNNs. It also facilitates a straightforward implementation of commonly used stochastic optimization algorithms. However, Bayes by Backprop also comes with a few drawbacks. The most noteworthy disadvantage is that the variational inference method assumes a considerably simplified version of the posterior distribution. Furthermore, it is implied in Equation 2.30, that the number of learnable parameters grows linearly because each model parameter is defined as a function of two distributional parameters.

2.3.2 Markov Chain Monte Carlo

As opposed to the Bayes by Backprop method, Markov Chain Monte Carlo (MCMC) [e.g. Richey, 2010] provides a more exact approach to perform Bayesian inference. The main idea of the MCMC method is to construct a Markov chain, with a certain transition probability $T(w^{(i+1)}|w^{(i)})$ of drawing the next sample $w^{(i+1)}$, which depends only on the previous sample $w^{(i)}$. Additionally, the transition also has to preserve the posterior distribution as a stationary distribution. Finally, with a sufficient amount of samples, the Markov chain converges to the exact posterior distribution of the model parameters $p(w|D)$. In general, the MCMC sampling algorithm is presented in Algorithm 2.

Algorithm 2 General MCMC algorithm

```
Set initial parameter values w^(0)
for i = 1 to N do
    Draw w_t given w^(i) according to the chosen proposal function
    Calculate acceptance probability α(w_t|w^(i)) according to Equation 2.32
    Draw a random number u ~ U[0, 1]
    if α(w_t|w^(i)) > u then
        w^(i+1) ← w_t
    else
        w^(i+1) ← w^(i)
    end if
end for
```

The transition probability that defines the Markov chain is a joint probability of the proposal $Q(w^{(i+1)}|w^{(i)})$ and the acceptance probability $\alpha(w^{(i+1)}|w^{(i)})$. The proposal function defines the movement from one sample to the next. Therefore, it is critical to choose a well-defined proposal function to ensure that the parameter space is well explored by the Markov chain. Several options for the proposal functions are presented in the following sections. Furthermore, the acceptance probability is important to preserve the stationarity of the posterior distribution. The acceptance probability is written mathematically as

$$\alpha(w^{(i+1)}|w^{(i)}) = \min\left(1, \frac{p(w^{(i+1)}|D)\, Q(w^{(i)}|w^{(i+1)})}{p(w^{(i)}|D)\, Q(w^{(i+1)}|w^{(i)})}\right). \tag{2.32}$$

Due to its ability to draw samples from an exact distribution, as well as its flexibility to choose between various proposal function definitions, MCMC is regarded as one of the

best sampling algorithms for inferring posterior distributions [e.g. Jospin et al., 2022]. However, there are also many difficulties associated with the implementation of MCMC. Most importantly, MCMC is a relatively expensive algorithm, since it requires an ample amount of samples to converge to the true distribution, especially in high-dimensional settings. Additionally, MCMC often suffers from inefficiency in the sampling process due to high autocorrelation.

Metropolis-Hastings

The Metropolis-Hastings (MH) algorithm [e.g. Chib and Greenberg, 1995] is one of the most commonly implemented MCMC algorithms due to its simple proposal function, which is defined as

$$w^{(i+1)} = w^{(i)} + \tau \cdot \epsilon, \tag{2.33}$$

where τ is the step size, and $\epsilon \sim \mathcal{N}(0, I)$. The proposal distribution, therefore, can be defined with

$$Q(w^{(i+1)}|w^{(i)}) = \exp(-||w^{(i+1)} - w^{(i)}||_2^2/\tau^2), \tag{2.34}$$

which is symmetric. As a consequence, $Q(w^{(i)}|w^{(i+1)}) = Q(w^{(i+1)}|w^{(i)})$, and Equation 2.32 reduces to

$$\alpha(w^{(i+1)}|w^{(i)}) = \min\left(1, \frac{p(w^{(i+1)}|D)}{p(w^{(i)}|D)}\right). \tag{2.35}$$

Despite its simplicity and nice properties, the efficiency of the MH algorithm is heavily dependent on the step size τ, and it scales terribly with increasing parameter dimension. As a solution, the adoption of gradient information has been proposed to alleviate this problem by improving parameter space exploration. Gradient-based MCMC also gained popularity in BNN applications because gradient information is easily accessible in neural network models.

Metropolis-Adjusted Langevin Algorithm

The Metropolis-Adjusted Langevin Algorithm (MALA) [e.g. Dwivedi et al., 2019] is a gradient-based MCMC algorithm that is inspired by the Langevin diffusion equation [e.g. Doob, 1942], which is used to describe a stochastic particle movement. The

proposal function takes a similar form with a stochastic gradient descent update:

$$w^{(i+1)} = w^{(i)} + \tau \nabla p(w^{(i)}|D) + \sqrt{2\tau\epsilon}, \tag{2.36}$$

where $\nabla p(w^{(i)}|D)$ is the gradient of the posterior distribution with respect to the model parameters.

Because gradient information is used in the proposal function, the proposal distribution is no longer symmetric. Therefore, the full form of Equation 2.32 has to be considered. The proposal distribution of MALA is defined as

$$Q(w^{(i+1)}|w^{(i)}) = \exp(-||w^{(i+1)} - w^{(i)} - \tau \nabla p(w^{(i)}|D)||_2^2/4\tau). \tag{2.37}$$

Although the computational demand of the algorithm is higher than the MH algorithm per individual step, the gradient information assists MALA so that it proposes more sampling movement to areas with higher posterior probability. Consequently, the convergence rate of MALA is better compared to the MH algorithm. Unfortunately, MALA is still sensitive to high step size.

The Barker Proposal

The Barker proposal [Livingstone and Zanella, 2022] is another MCMC algorithm that uses gradient information to influence the direction in which the sampling is performed. Mathematically, the proposal function is written as

$$w^{(i+1)} = w^{(i)} + b \cdot \epsilon, \tag{2.38}$$

where $b = 1$ with the probability of $p = 1/(1 + \exp(-\epsilon \nabla p(w^{(i)}|D)))$, and $b = -1$ otherwise. Here, the probability p is high if the proposed direction is in agreement with the direction of the posterior gradient. Consequently, the algorithm pushes the sampling direction more frequently towards regions with higher posterior probability.

The distribution of the Barker proposal is also not symmetric due to the implementation of the gradient, and the full form of Equation 2.32 should be considered to calculate

the acceptance probability. The Barker proposal distribution is defined as

$$Q(w^{(i+1)}|w^{(i)}) = \frac{1}{1 + \exp\left((w^{(i)} - w^{(i+1)})\nabla p(w^{(i)}|D)\right)}. \tag{2.39}$$

Because the gradient information does not influence the step size in the Barker proposal, it improves the robustness of tuning parameters and provides better scalability compared to MALA.

3 Contributions

This chapter discusses and answers the research questions posed in this thesis. Each research question is analyzed in a separate section, which summarizes published works that contribute to answering the related research question. Section 3.1 considers the contribution of the first publication to address the question of exploiting prior physical knowledge for dynamic systems modeling. Section 3.2 examines the second and third publications, focusing on increasing the complexity of the first research question, to model spatio-temporal systems. Finally, Section 3.3 investigates the fourth publication, concentrating on the question about uncertainty quantification in the proposed neural network model.

3.1 Contribution 1: Improving Prediction and Plausibility of a Dynamic Thermochemical Energy Storage System

The first publication [Praditia et al., 2020] focuses on developing a physically consistent and reliable hybrid model for dynamic systems. One particularly intriguing example of such a system is a Thermochemical Energy Storage (TCES) system, more specifically the variant with calcium oxide (CaO)/calcium hydroxide ($Ca(OH)_2$) [e.g. Schaube et al., 2013]. TCES is a fascinating application because of its potential to develop into a relatively cheap and efficient energy storage system. However, the physical laws governing the TCES behavior are complex and, in turn, induce difficulties in the operational control of such a system.

To tackle this issue, a model that is suitable to treat dynamic systems is required. The Nonlinear Autoregressive Network with Exogeneous Inputs (NARX) is one appropriate

example because its recurrent structure allows not only learning dependency between variables at adjacent time steps, but also over larger time lags. Additionally, NARX takes the influence of external driving forces on the system's internal states into account, in the form of exogenous inputs. It is also crucial to encode the recurrence in the training algorithm, for the model to better capture the long-term dynamics. More importantly, rigorous regularization is employed in the proposed modeling framework. The regularization has two distinct purposes: the L2 regularization imposes the model to be as simple as possible, and the physics-based regularization constrains the model to obey known physical laws.

First, the advantage of multiple feedback delays in the NARX model is tested by comparing its prediction with a NARX with a single feedback delay, which is equivalent to a regular Recurrent Neural Network (RNN). It is shown that using the predictions from multiple preceding time steps as input improves the performance of the NARX model, as opposed to using only a regular RNN. It is also important to note that this is beneficial only up to a certain extent, after which there is no longer notable performance improvement. Additionally, using too many feedback delays results in increasing computational burden, especially in the backpropagation process.

Second, the enforcement of recurrence during the training process is shown to be highly advantageous for long-term predictions. Models that are trained in a closed-loop structure, or in other words, without using teacher forcing, produce significantly more accurate and numerically stable predictions. This can be attributed to the fact that, when the models are trained in an open-loop structure, they do not learn to minimize the error accumulation in predictions over a longer time horizon.

Third, the model that is trained with regularization terms is compared with the model that is trained without them. Here, the physical regularization is formulated based on conservation laws, such as mass and energy conservation, as well as the known monotonic behavior of the predicted internal states and the strict positivity of some predictions that are in the form of fractions. It is shown that the combination of this physical regularization and the L2 regularization generates the best model. The resulting model is physically consistent even in its worst predictions.

3.2 Contribution 2: Learning Unknown Constituents of Advection-Diffusion-Reaction Equations Using the Finite Volume Neural Network

The second publication [Praditia et al., 2021] and the third publication [Karlbauer et al., 2022] add a layer of complexity by including the spatial dimension into the equation. More specifically, this contribution focuses on advection-diffusion-reaction equations, which are Partial Differential Equations (PDEs) of the first order in time and second order in space. Existing machine learning models that were proposed to solve PDEs either rely too much on data, causing them to be physically inconsistent, or too much on simplified physical equations with inappropriate assumptions. The aim of this contribution is, therefore, to answer the second research question, which is to develop a physically consistent and reliable hybrid model for spatio-temporal problems. Additionally, it addresses the proper treatment of boundary conditions, which is critical in obtaining a unique solution of PDEs, but is often improperly administered in existing machine learning models. Furthermore, the existing models are mostly developed only as faster surrogate models with inadequate physical interpretability. In contrast, this contribution aims to learn unknown constitutive and closure relationships that are often not fully understood, and improve upon the existing scientific hypothesis to accelerate further scientific discoveries.

This thesis proposes the Finite Volume Neural Network (FINN), which adopts the Finite Volume Method (FVM) discretization structure, fused with the learning capability of ANN. Here, the FVM structure serves as prior physical knowledge that regularizes the model. The used knowledge states that the physical behavior is translation invariant, and that fluxes and balancing are meaningful concepts. More importantly, the FVM allows for a convenient approach to incorporating different types of boundary conditions into the model. To enable gradient calculations on FINN, a differentiable ODE solver in the form of the Neural ODE method is implemented. In addition to its differentiability, the Neural ODE method provides a means to use a higher-order time integration method for better accuracy and numerical stability. FINN is also modular, meaning that different parts that compose FINN are responsible for learning different parts of the modeled partial differential equations. As a consequence, specific modules can be assigned to learn unknown closure/constitutive equations from the available

training data, and the resulting learned relations are physically interpretable.

To show these contributions, FINN is first compared against various state-of-the-art deep learning models, including purely data-driven models as well as physically-motivated deep learning models. It is shown that the purely data-driven models are not capable of learning only from scarce training data. Additionally, while most physically-motivated models perform comparably during training, FINN notably outperforms all other models by a few orders of magnitude when tested on data that are generated with different boundary conditions, clearly demonstrating the advantage of properly incorporating the boundary conditions into the model. Not only is FINN more accurate, but it also produces extremely consistent predictions, as evidenced by the narrow confidence interval of its predictions, after training with multiple random initializations.

Second, the higher-order integration method implemented in the Neural ODE framework is also shown to improve numerical stability during the model training. The training of FINN when combined with the explicit Euler method is severely unstable, only able to reach several epochs before the error accumulates and explodes.

Third, and arguably the most important aspect, is the interpretability of FINN. By adopting a modular structure, the different modules that construct FINN are physically interpretable. This allows not only to extract the learned parameters like the diffusion coefficient, but also the learned functions such as the advective velocity in Burgers' equation, the retardation factor in the diffusion-sorption equation, and the reaction function in the diffusion-reaction and Allen-Cahn equations. It is also demonstrated that the learned functions approximate the ground truths that are used to generate synthetic datasets. Furthermore, FINN can learn the unknown retardation factor function from sparse laboratory measurement data, showing its freedom and flexibility to learn, independent of existing parametric equations.

3.3 Contribution 3: Uncertainty Quantification on Groundwater Contaminant Experimental Data

Observation data on natural systems are often scarce due to the complicated and expensive nature of the measurement process, both in a controlled environment such as

a laboratory and even more in the uncontrolled settings of field sites. Additionally, the obtained measurement data are noisy, leading to more uncertainty about the correctness and quality of the data itself. Consequently, this leads to a questionable quality of the model that is trained on these data. Therefore, it is of extreme importance to develop a model that is robust to process real-world measurement data that possesses those inherent characteristics, especially to assist decision-making processes in safety-critical problems and to be honest about the confidence of newly-learned scientific hypotheses. The fourth publication [Praditia et al., 2022] considers this issue by investigating the effectiveness of various uncertainty quantification methods for FINN.

One way to perform uncertainty quantification on FINN is to reformulate it as a Bayesian Neural Network (BNN), where its parameters are random variables instead of deterministic. One of the most popular algorithms to train BNNs is the Bayes by Backprop method. The Bayes by Backprop method is a variational inference method, where each random variable is parameterized using a mean and a standard deviation value. These mean and standard deviation values are then adjusted to fit the model predictions to the training data. Apart from the Bayes by Backprop method, a BNN can be trained using sampling algorithms such as Markov Chain Monte Carlo (MCMC). In MCMC methods, the distribution of the random variables is not parameterized but is rather sampled from the exact posterior distribution. Moreover, MCMCs sample the random variables by employing a transition probability, which is analogous to gradient-based optimization algorithms used for training ANNs. Therefore, this thesis also explores gradient-based MCMC algorithms to improve sampling convergence.

A sparse laboratory measurement dataset on a contaminant diffusion-sorption process in clay is used to demonstrate the robustness of the uncertainty quantification methods that are adopted in FINN. The Bayes by Backprop method fails to properly parameterize FINN's random parameters. Initialization of the standard deviation parameters with higher values results in difficulties in the training process, since high standard deviation values lead to higher noise in the sampled parameter values. On the other hand, initialization of the standard deviation parameters with lower values results in the model being too deterministic, because the standard deviation values are not updated properly during training. Additionally, the variational parameterization adopted in Bayes by Backprop signifies that the resulting random parameter distributions are assumed to be normal, even though ANN parameters are unlikely to follow a normal distribution after a non-linear Bayesian inference.

To mitigate this issue, this thesis applies various MCMC sampling algorithms to identify the distribution of FINN's parameters. It is shown that gradient-based MCMC algorithms, namely the Metropolis-Adjusted Langevin Algorithm (MALA) and the Barker proposal, provide a better convergence rate compared to the plain random walk Metropolis-Hastings (MH) algorithm. Furthermore, this thesis proposes to train FINN deterministically before initiating the MCMC algorithm, to provide a better starting point for the sampling chains. It is shown that a better starting point massively reduces the required runtime of the sampling algorithm. Additionally, the evidence presented in this thesis points to the Barker proposal as the most suitable MCMC method for FINN.

Finally, the samples of FINN's parameters obtained using the Barker MCMC algorithm are used to provide a confidence interval of FINN's prediction. They are compared against the prediction obtained by a calibrated, conventional physical model. This calibrated physical model is shown to be inaccurate when tested on different soil samples. FINN outperforms the physics-based model both during training and testing. This can be attributed to the fact that FINN is capable of learning a shape-free retardation factor function, whereas the physics-based model is limited to discrete choices of parametric equations. More importantly, the confidence interval provided by FINN's predictions covers most of the noisy measurement data, showing that FINN sufficiently identifies multiple hypotheses with similar likelihoods to explain the noisy data.

4 Conclusion and Outlook

The description of naturally occurring physical phenomena by existing scientific models is still imperfect, especially at the scale that is critical for decision-making processes. The recent growth of data-driven methods can potentially improve our understanding of more complex physical processes. However, these methods heavily depend on the availability and quality of observation data, which are often difficult to obtain. On the other hand, physics-informed models follow rigid physical constraints that are too prohibitive, hence hindering their ability to learn useful new insights. This thesis addresses this issue, by proposing an uncertainty-aware hybrid modeling framework to assist in scientific discovery, and ultimately to establish a continuous improvement of scientific knowledge. This chapter summarizes the findings of this thesis to answer the research questions and provides an outlook and presents ideas for future research.

4.1 Conclusion

To reach the goal of this thesis, three research questions had been posed. In this section, the contributions of this thesis are compiled to answer the corresponding research questions.

Research Question 1: How can prior physical knowledge be adopted to design a consistent and reliable hybrid model for dynamic systems?

This thesis proposes several strategies to answer this question. First, a recurrent structure is required to model dynamic systems, to encode the temporal dependence between the unknown states at different points in time. Correspondingly, the recurrence also has to be enforced in the training process to teach the model to regulate the error

accumulation over time. Second, multiple feedback delays are also beneficial to teach the model to learn about long-term autocorrelation. Finally, a physics-based regularization imposes the physical plausibility of the model prediction. The first publication demonstrates the importance of these aspects with an application in the Thermochemical Energy Storage (TCES) system. It is also important to note that, although the framework is demonstrated in the TCES system, it is generally applicable to other dynamic systems.

Research Question 2: How can prior physical and numerical knowledge be adopted to design a consistent and reliable hybrid model for dynamic and spatially distributed systems?

The second and third publications propose answers to this research question. To model spatio-temporal problems that can be written mathematically as Partial Differential Equations (PDEs), a well-defined spatial discretization strategy is essential. This can be realized in the form of the Finite Volume Method (FVM), which serves as the fundamental structure of the proposed hybrid model. The FVM structure is then complemented by learnable modules that enable data-driven discovery. Additionally, the structure allows for the proper treatment of different types of numerical boundary conditions, further enhancing the physical consistency of the model. To ensure numerical stability during training, a higher-order ODE solver that is implemented in the Neural ODE method is employed. In the corresponding publications, numerical experiments are conducted on various application domains such as Burgers' equation, the diffusion-sorption equation, the diffusion-reaction equation, and the Allen-Cahn equation. This shows the broad sphere of application of the proposed Finite Volume Neural Network (FINN) model.

Research Question 3: How can the hybrid model learn about its own total (predictive) uncertainty in a computationally effective manner, so that it is appropriate for real-world applications or could facilitate scientific hypothesis testing?

The fourth publication proposes answers to the third research question. First, the Markov Chain Monte Carlo (MCMC) method is preferable over the Bayes by Backprop

method for inferring the random parameter distributions of a Bayesian Neural Network (BNN). MCMC allows for a direct sampling from the exact posterior distribution, whereas Bayes by Backprop limits the posterior to a Gaussian distribution. Second, gradient-based MCMC algorithms improve the convergence rate of the sampling in comparison to the plain random walk Metropolis-Hastings algorithm. Third, a BNN model has to possess a relatively low number of parameters to ensure flexibility in choosing between different Bayesian inference methods and feasibility in implementing the chosen method while taking the available computing power into account. This thesis shows the implementation of these aspects through a Bayesian FINN, applied to a groundwater contaminant transport problem. When trained with sparse laboratory measurement data, the Bayesian FINN outperforms a calibrated, conventional physical model and captures the noisy measurement data nicely within its confidence interval.

4.2 Outlook

Although the results presented in this thesis show promising potential, there are still open questions that need to be addressed to bridge the gap between the current state-of-the-art and the ultimate goal. First, FINN is so far only implemented on spatial domains with regular grid shapes and homogeneous properties. While this might be a suitable solution/assumption for applications on a lab scale, it might not be appropriate for applications on a larger field scale. This is especially critical for scientific discovery in physical systems that are difficult to simulate in a lab-scale experiment. One way to incorporate spatial heterogeneity and irregular grids into the hybrid model are by employing a message-passing graph neural network structure [Gilmer et al., 2017, Brandstetter et al., 2022]. With this structure, each node can learn its distinct parameter instead of a single homogenous parameter that is used for the whole domain. Furthermore, geostatistical methods can also be employed as additional constraints to regulate a physically-plausible spatial heterogeneity. A graph model structure also allows learning of the numerical discretization stencils through its message-passing feature.

Second, the current implementation of FINN is not optimized for efficient computation. For larger-scale applications, which require FINN to process a large amount of data, the computational time could pose a problem. To perform computations at such

a large scale, parallel computing might be necessary, and therefore, an optimized implementation of FINN is required for both CPU and GPU computations. One major source of the bottleneck is the need to perform sequential iterations to integrate the learned equation over the whole temporal domain. One possible solution is to partition the temporal domain into different segments. These segments or batches can then be computed in parallel to improve computational efficiency.

Third, it is often difficult to define a precise boundary on a field-scale observation. More concretely, the domain of interest is often treated as a separate entity even though it might be influenced by a connected neighboring domain. For example, the weather in Germany is influenced by the weather in the neighboring countries. However, it is not feasible to perform a full weather simulation on all the neighboring countries. To eliminate the need of conducting these expensive simulations, boundary conditions are defined to encode the influence of the weather from the neighboring countries on the weather in Germany. Consequently, boundary condition learning becomes an important problem. The first steps have been taken to employ FINN to perform this task [Horuz et al., 2022], and further work on more complicated boundary condition types is still necessary.

Fourth, performing uncertainty quantification on FINN, and ANNs in general, is notorious for its difficulty due to the statistical unidentifiability of ANNs [Jospin et al., 2022]. In other words, the parameters of ANNs are non-unique, because of the weight symmetry inside a hidden layer. This means that an exchange of two hidden nodes inside a hidden layer, followed by the permutation of the corresponding weights connected to these hidden nodes, leads to the ANN producing the same output. A possible workaround for this issue is to define a stronger prior with a narrower distribution, to restrict the exploration space of the sampling algorithms. Furthermore, a more rigorous and complete uncertainty quantification analysis also needs to be performed on FINN. More specifically, uncertainty quantification can be performed on multiple FINN models with different concepts and structures, when the form of the governing PDE is still uncertain.

Finally, FINN's applicability to real-world data has only been tested on the groundwater contaminant transport problem. More real-world applications are necessary not only to further validate its versatility but also to contribute to discoveries in broader scientific areas. Examples of such applications are sea surface temperature prediction, climate

modeling, multi-phase fluid flow in porous media, or even a larger-scale groundwater contamination problem.

Bibliography

M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. doi: 10.1016/j.inffus.2021.05.008.

C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, 2018. doi: 10.1007/978-3-319-94463-0.

T. I. Allen, J. D. Griffin, M. Leonard, D. J. Clark, and H. Ghasemi. The 2018 national seismic hazard assessment of Australia: Quantifying hazard changes and model uncertainties. *Earthquake Spectra*, 36:5–43, 2020. doi: 10.1177/8755293019900777.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1–2):5–43, 2003. doi: 10.1023/A:1020281327116.

T. Appenzeller. The scientists' apprentice. *Science*, 357(6436):16–17, 2017. doi: 10.1126/science.357.6346.16.

G. Arfken, H. Weber, and F. Harris. *Mathematical Methods for Physicists: A Comprehensive Guide*. Elsevier Science, 2013. doi: 10.1016/C2009-0-30629-7.

R. Ashena and G. Thonhauser. *Application of Artificial Neural Networks in Geoscience and Petroleum Industry*, pages 127–166. Springer International Publishing, 2015. doi: 10.1007/978-3-319-16531-8_4.

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. 2018. doi: 10.48550/arXiv.1803.01271.

J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber. Evolving memory cell structures for sequence learning. In *Artificial Neural Networks – ICANN 2009*, pages 755–764, 2009. doi: 10.1007/978-3-642-04277-5_76.

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

A. Benkirane and A. Touzani. *Partial Differential Equations*. Lecture Notes in Pure and Applied Mathematics. Taylor & Francis, 2002. doi: 10.1201/9780203910108.

K. J. Bergen, P. A. Johnson, M. V. de Hoop, and G. C. Beroza. Machine learning for data-driven discovery in solid earth geoscience. *Science*, 363(6433):eaau0323, 2019. doi: 10.1126/science.aau0323.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, page 1613–1622, 2015. doi: 10.48550/arXiv.1505.05424.

I. C. Bourg and G. Sposito. Connecting the molecular scale to the continuum scale for diffusion processes in Smectite-rich porous media. *Environmental Science & Technology*, 44(6):2085–2091, 2010. doi: 10.1021/es903645a.

Z. Boussaada, O. Curea, A. Remaci, H. Camblong, and N. Mrabet Bellaaj. A nonlinear autoregressive exogenous (NARX) neural network model for the prediction of the daily direct solar radiation. *Energies*, 11(3), 2018. doi: 10.3390/en11030620.

R. N. Bracewell and R. N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill New York, 1986.

J. Brandstetter, D. E. Worrall, and M. Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2202.03376.

L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001. doi: 10.1214/ss/1009213726.

G. H. Brown, M. C. Brooks, A. L. Wood, M. D. Annable, and J. Huang. Aquitard contaminant storage and flux resulting from dense nonaqueous phase liquid source

zone dissolution and remediation. *Water Resources Research*, 48(6), 2012. doi: https://doi.org/10.1029/2011WR011141.

J. Buitrago and S. Asfour. Short-term forecasting of electric loads using nonlinear autoregressive artificial neural networks with exogenous vector inputs. *Energies*, 10 (1), 2017. doi: 10.3390/en10010040.

J. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008. doi: 10.1002/9781119121534.

G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová . Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4), 2019. doi: 10.1103/revmodphys.91.045002.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018. doi: 10.48550/arXiv.1806.07366.

S. Chen, S. Billings, and P. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990. doi: 10.1080/00207179008934126.

S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995. doi: 10.2307/2684568.

P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong. CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022. doi: https://doi.org/10.1016/j.cma.2022.114909.

K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014. doi: 10.3115/v1/W14-4012.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning and Representation Learning*, 2014. doi: 10.48550/arXiv.1412.3555.

R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967. doi: 10.1147/rd.112.0215.

H. Crowley, D. Rodrigues, V. Silva, V. Despotaki, L. Martins, X. Romão, J. Castro, N. Pereira, A. Pomonis, A. Lemoine, A. Roullé, B. Tourliere, G. Weatherill, K. Pitilakis, E. Zurich, S. Correia, S. Akkar, U. Hancilar, and P. Covi. The European Seismic Risk Model 2020 (ESRM 2020). In *2nd International Conference on Natural Hazards & Infrastructure*, 2019.

J. L. Doob. The Brownian Movement and Stochastic Equations. *Annals of Mathematics*, 43(2):351–369, 1942.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. 2017. doi: 10.48550/arXiv.1702.08608.

K. Du and M. Swamy. *Neural Networks and Statistical Learning*. Springer London, 2013. doi: 10.1007/978-1-4471-7452-3.

R. Dwivedi, Y. Chen, M. J. Wainwright, and B. Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast. *Journal of Machine Learning Research*, 20(183):1–42, 2019. doi: 10.48550/arXiv.1801.02309.

R. Dybowski. Interpretable machine learning as a tool for scientific discovery in chemistry. *New Journal of Chemistry*, 44:20914–20920, 2020. doi: 10.1039/D0NJ02592E.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. doi: 10.1016/0364-0213(90)90002-E.

J. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-56026-2.

T. Freiesleben, G. König, C. Molnar, and A. Tejero-Cantero. Scientific inference with interpretable machine learning: Analyzing models to learn about real-world phenomena. 2022. doi: 10.48550/arXiv.2206.05487.

S. Friedlander, S. Friedlander, and D. Serre. *Handbook of Mathematical Fluid Dynamics*. Elsevier Science, 2002.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 1263–1272, 2017. doi: 10.48550/arXiv.1704.01212.

E. Goan and C. Fookes. Bayesian Neural Networks: An Introduction and Survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer International Publishing, 2020. doi: 10.1007/978-3-030-42553-1_3.

I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. MIT Press, 2016.

V. Guen and N. Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11474–11484, 2020. doi: 10.1109/CVPR42600.2020.01149.

H. Gupta and G. Nearing. Debates–The future of hydrological sciences: A (common) path forward? Using models and data to learn: A systems theoretic perspective on the future of hydrological science. *Water Resources Research*, 50(6):5351–5359, 2014. doi: 10.1002/2013WR015096.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, 2009. doi: 10.1007/978-0-387-84858-7.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Q. He, D. Barajas-Solano, G. Tartakovsky, and A. M. Tartakovsky. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141:103610, 2020. doi: 10.1016/j.advwatres.2020.103610.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9: 1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*, pages 237–243. 2001. doi: 10.1109/9780470544037.ch14.

J. Hoffmann, Y. Bar-Sinai, L. M. Lee, J. Andrejevic, S. Mishra, S. M. Rubinstein, and C. H. Rycroft. Machine learning in a data-limited regime: Augmenting experiments with synthetic data uncovers order in crumpled sheets. *Science Advances*, 5(4): eaau6792, 2019. doi: 10.1126/sciadv.aau6792.

Z. Hongren. A comparative study of numerical solution of PDE in geosciences. *Acta Geologica Sinica - English Edition*, 7(1):83–93, 1994. doi: 10.1111/j.1755-6724.1994.mp7001007.x.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. doi: 10.1016/0893-6080(91)90009-T.

C. C. Horuz, M. Karlbauer, T. Praditia, M. V. Butz, S. Oladyshkin, W. Nowak, and S. Otte. Inferring boundary conditions in finite volume neural networks. In *ICANN 2022: 31st International Conference on Artificial Neural Networks*, 2022. doi: 10.1007/978-3-031-15919-0_45.

K. Hsu, X. Gao, and H. Sorooshian, S. Gupta. Precipitation estimation from remotely sensed information using artificial neural networks. *Journal of Applied Meteorology*, 36:1176–1190, 1995a. doi: 10.1029/95WR01955.

K. Hsu, H. Gupta, and S. Sorooshian. Artificial neural network modeling of the rainfall-runoff process. *Water Resources Research*, 31:2517–2530, 1995b. doi: 10.1029/95WR01955.

Y. Huang, Y. Tang, H. Zhuang, J. VanZwieten, and L. Cherubin. Physics-informed tensor-train ConvLSTM for volumetric velocity forecasting of the loop current. *Frontiers in Artificial Intelligence*, 2021. doi: 10.3389/frai.2021.780271.

IPCC. *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2013.

E. Isaacson and H. Keller. *Analysis of Numerical Methods*. Dover Publications, 1994.

X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics Guided RNNs for Modeling Dynamical Systems: A Case Study in Simulating Lake Temperature Profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, pages 558–566. doi: 10.1137/1.9781611975673.63.

P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, 2020. doi: 10.1016/j.neunet.2020.08.017.

M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015. doi: 10.1126/science.aaa8415.

L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun. Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022. doi: 10.1109/MCI.2022.3155327.

J. M. Joyce. *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-04898-2_327.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. doi: 10.1115/1.3662552.

M. Karlbauer, S. Otte, H. Lensch, T. Scholten, V. Wulfmeyer, and M. Butz. A distributed neural network architecture for robust non-linear spatio-temporal prediction. 2019. doi: 10.48550/arXiv.1912.11141.

M. Karlbauer, T. Praditia, S. Otte, S. Oladyshkin, W. Nowak, and M. V. Butz. Composing partial differential equations with physics-aware neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 10773–10801, 2022. doi: 10.48550/arXiv.2111.11798.

G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021. doi: 10.1038/s42254-021-00314-5.

A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017. doi: 10.1109/tkde.2017.2720168.

A. Karpatne, W. Watkins, J. Read, and V. Kumar. *Physics-Guided Neural Networks (PGNN): An Application in Lake Temperature Modeling*. Chapman and Hall/CRC, 2022. doi: 10.1201/9781003143376.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. doi: 10.48550/arXiv.1412.6980.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014. doi: 10.48550/arXiv.1312.6114.

J. Koch and W. Nowak. Predicting DNAPL mass discharge and contaminated site longevity probabilities: Conceptual model and high-resolution stochastic simulation. *Water Resources Research*, 51(2):806–831, 2015. doi: 10.1002/2014WR015478.

A. Lavin, H. Zenil, B. Paige, D. Krakauer, J. Gottschlich, T. Mattson, A. Anandkumar, S. Choudry, K. Rocki, A. G. Baydin, C. Prunkl, B. Paige, O. Isayev, E. Peterson, P. L. McMahon, J. Macke, K. Cranmer, J. Zhang, H. Wainwright, A. Hanuka, M. Veloso, S. Assefa, S. Zheng, and A. Pfeffer. Simulation intelligence: Towards a new generation of scientific methods. 2021. doi: 10.48550/arXiv.2112.03235.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. *Object recognition with gradient-based learning*, pages 319–345. Springer, 1999. doi: 10.1007/3-540-46805-6_19.

Y. Li, L. Song, S. Zhang, L. Kraus, T. Adcox, R. Willardson, A. Komandur, and N. Lu. A TCN-based hybrid forecasting framework for hours-ahead utility-scale PV forecasting. 2021a. doi: 10.48550/arXiv.2111.08809.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021b. doi: 10.48550/arXiv.2010.08895.

Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. 2021c. doi: 10.48550/arXiv.2111.03794.

J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.

S. Livingstone and G. Zanella. The Barker proposal: combining robustness and efficiency in gradient-based MCMC. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(2):496—523, 2022. doi: 10.1111/rssb.12482.

D. Logan. *A First Course in the Finite Element Method*. PWS-Kent Publishing Company, 1992.

Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 3276–3285, 2018. doi: 10.48550/arXiv.1710.10121.

D. J. C. Mackay. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995. doi: 10.1088/0954-898X_6_3_011.

H. Maier and G. Dandy. The use of artificial neural networks for the prediction of water quality parameters. *Water Resources Research*, 32(4):1013–1022, 1996. doi: 10.1029/96WR03529.

G. Marchuk. *Numerical Methods in Weather Prediction*. Elsevier, 1974. doi: 10.1016/B978-0-12-470650-7.X5001-4.

S. Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in Big Data*, 4:669097, 2021. doi: 10.3389/fdata.2021.669097.

T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. doi: 10.1016/j.artint.2018.07.007.

S. Molins and P. Knabner. Multiscale approaches in reactive transport modeling. *Reviews in Mineralogy & Geochemistry*, 85(1):27–48, 2019. doi: 10.2138/rmg.2019.85.2.

K. Morton and D. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, 1994. doi: 10.1017/CBO9780511812248.

F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer Cham, 2016. doi: 10.1007/978-3-319-16874-6.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019. doi: 10.1073/pnas.1900654116.

W. Nowak and A. Guthke. Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, 18(11), 2016. doi: 10.3390/e18110409.

C. Olah. Understanding LSTM Networks, 2015. URL `https://colah.github.io/posts/2015-08-Understanding-LSTMs`.

F. Oviedo, J. L. Ferres, T. Buonassisi, and K. T. Butler. Interpretable and explainable machine learning for materials science and chemistry. *Accounts of Materials Research*, 3(6):597–607, 2022. doi: 10.1021/accountsmr.1c00244.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1310–1318, 2013. doi: 10.48550/arXiv.1211.5063.

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, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019. doi: 10.48550/arXiv.1912.01703.

J. Peinado, J. Ibáñez, E. Arias, and V. Hernández. Adams–Bashforth and Adams–Moulton methods for solving differential Riccati equations. *Computers Mathematics with Applications*, 60(11):3032–3045, 2010. doi: 10.1016/j.camwa.2010.10.002.

J. Petch, S. Di, and W. Nelson. Opening the black box: The promise and limitations of explainable machine learning in cardiology. *Canadian Journal of Cardiology*, 38 (2):204–213, 2022. doi: 10.1016/j.cjca.2021.09.004.

L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishechenko. *Mathematical Theory of Optimal Processes*. Classics of Soviet Mathematics. John Wiley & Sons, 1987. doi: 10.1201/9780203749319.

T. Praditia, T. Walser, S. Oladyshkin, and W. Nowak. Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture. *Energies*, 13(15):3873, 2020. doi: 10.3390/en13153873.

T. Praditia, M. Karlbauer, S. Otte, S. Oladyshkin, M. Butz, and W. Nowak. Finite volume neural network: Modeling subsurface contaminant transport. 2021. doi: 10.48550/arXiv.2104.06010.

T. Praditia, M. Karlbauer, S. Otte, S. Oladyshkin, M. V. Butz, and W. Nowak. Learning groundwater contaminant diffusion-sorption processes with a finite volume neural network. *Water Resources Research*, 58(12):e2022WR033149, 2022. doi: 10.1029/2022WR033149.

C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. Universal differential equations for scientific machine learning. 2020. doi: 10.48550/arXiv.2001.04385.

M. Raissi, P. Perdikaris, and 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*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

J. S. Read, X. Jia, J. Willard, A. P. Appling, J. A. Zwart, S. K. Oliver, A. Karpatne, G. J. A. Hansen, P. C. Hanson, W. Watkins, M. Steinbach, and V. Kumar. Process-guided deep learning predictions of lake water temperature. *Water Resources Research*, 55(11):9173–9190, 2019. doi: 10.1029/2019WR024922.

M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, and Prabhat. Deep learning and process understanding for data-driven Earth system science. *Nature*, 566(7743):195–204, 2019. doi: 10.1038/s41586-019-0912-1.

M. Richey. The evolution of Markov chain Monte Carlo methods. *The American Mathematical Monthly*, 117(5):383–413, 2010. doi: 10.4169/000298910x485923.

J. Riebesell. Random TikZ Collection, 2021. URL `https://github.com/janosh/tikz`.

R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke. Explainable machine learning for scientific insights and discoveries. *IEEE Access*, 8:42200–42216, 2020. doi: 10.1109/ACCESS.2020.2976199.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

I. H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, 2021. doi: 10.1007/s42979-021-00815-1.

F. Schaube, A. Kohzer, J. Schütz, A. Wörner, and H. Müller-Steinhagen. De- and rehydration of $Ca(OH)_2$ in a reactor with direct heat transfer for thermo-chemical heat storage. Part A: Experimental results. *Chemical Engineering Research and Design*, 91(5):856–864, 2013. doi: 10.1016/j.cherd.2012.09.020.

A. Schöniger, T. Wöhling, L. Samaniego, and W. Nowak. Model selection on solid ground: Rigorous comparison of nine ways to evaluate Bayesian model evidence. *Water Resources Research*, 50(12):9484–9513, 2014. doi: 10.1002/2014WR016062.

B. Sengupta, K. J. Friston, and W. D. Penny. Gradient-based MCMC samplers for dynamic causal modelling. *NeuroImage*, 125:1107–1118, 2016. doi: 10.1016/j.neuroimage.2015.07.043.

A.-N. Sharkawy. Principle of neural network and its main types: Review. *Journal of Advances in Applied Computational Mathematics*, 7:8–19, 2020. doi: 10.15377/2409-5761.2020.07.2.

X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, 2015. doi: 10.48550/arXiv.1506.04214.

A. H. Stroud. *Initial Value Problems for Ordinary Differential Equations*, pages 207–303. Springer New York, 1974. doi: 10.1007/978-1-4612-6390-6_4.

A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano. Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5):e2019WR026731. doi: 10.1029/2019WR026731.

A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952. doi: 10.1098/rstb.1952.0012.

A. Tutueva, T. Karimov, and D. Butusov. Semi-implicit and semi-explicit Adams-Bashforth-Moulton methods. *Mathematics*, 8(5), 2020. doi: 10.3390/math8050780.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. doi: 10.48550/arXiv.1706.03762.

P. Veličković. 2D Convolution, 2016. URL `https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution`.

H. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 1995.

P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

P. Westermann, M. Welzel, and R. Evins. Using a deep temporal convolutional network as a building energy surrogate model that spans multiple climate zones. *Applied Energy*, 278:115563, 2020. doi: 10.1016/j.apenergy.2020.115563.

R. T. Wetherald and S. Manabe. Simulation of hydrologic changes associated with global warming. *Journal of Geophysical Research: Atmospheres*, 107(D19), 2002. doi: 10.1029/2001JD001195.

J. Woessner, D. Laurentiu, D. Giardini, H. Crowley, F. Cotton, Grünthal, G. Valensise, R. Arvidsson, R. Basili, M. B. Demircioglu, S. Hiemer, C. Meletti, R. W. Wusson, A. N. Rovida, K. Sesetyan, M. Stucchi, and T. S. Consortium. The 2013 European seismic hazard model: Key components and results. *Bulletin of Earthquake Engineering*, 13:3553–3596, 2015. doi: 10.1007/s10518-015-9795-1.

List of Publications

Publication 1: T. Praditia, T. Walser, S. Oladyshkin, and W. Nowak. Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture. *Energies*, 13(15):3873, 2020. doi: 10.3390/en13153873.

Publication 2: T. Praditia, M. Karlbauer, S. Otte, S. Oladyshkin, M. V. Butz, and W. Nowak. Finite volume neural network: Modeling subsurface contaminant transport. In *Deep Learning for Simulation, International Conference on Learning Representations 2021 Workshop*, 2021. doi: 10.48550/arXiv.2104.06010.

Publication 3: M. Karlbauer, T. Praditia, S. Otte, S. Oladyshkin, W. Nowak, and M. V. Butz. Composing partial differential equations with physics-aware neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 10773-10801, 2022. doi: 10.48550/arXiv.2111.11798.

Publication 4: T. Praditia, M. Karlbauer, S. Otte, S. Oladyshkin, M. V. Butz, and W. Nowak. Learning groundwater contaminant diffusion-sorption processes with a finite volume neural network. *Water Resources Research*, 58(12), 2022. doi: 10.1029/2022WR033149.

A Publication 1: Improving Thermochemical Energy Storage Dynamics Forecast with Physics-Inspired Neural Network Architecture

energies

MDPI

Article

Improving Thermochemical Energy Storage Dynamics Forecast with Physics-Inspired Neural Network Architecture

Timothy Praditia *, Thilo Walser, Sergey Oladyshkin and Wolfgang Nowak

Department of Stochastic Simulation and Safety Research for Hydrosystems, Institute for Modelling Hydraulic and Environmental Systems, Universität Stuttgart, Pfaffenwaldring 5a, 70569 Stuttgart, Germany; thilo-walser@web.de (T.W.); sergey.oladyshkin@iws.uni-stuttgart.de (S.O.); wolfgang.nowak@iws.uni-stuttgart.de (W.N.)

* Correspondence: timothy.praditia@iws.uni-stuttgart.de

Received: 23 June 2020; Accepted: 24 July 2020; Published: 29 July 2020

Abstract: Thermochemical Energy Storage (TCES), specifically the calcium oxide (CaO)/calcium hydroxide ($Ca(OH)_2$) system is a promising energy storage technology with relatively high energy density and low cost. However, the existing models available to predict the system's internal states are computationally expensive. An accurate and real-time capable model is therefore still required to improve its operational control. In this work, we implement a Physics-Informed Neural Network (PINN) to predict the dynamics of the TCES internal state. Our proposed framework addresses three physical aspects to build the PINN: (1) we choose a Nonlinear Autoregressive Network with Exogeneous Inputs (NARX) with deeper recurrence to address the nonlinear latency; (2) we train the network in closed-loop to capture the long-term dynamics; and (3) we incorporate physical regularisation during its training, calculated based on discretized mole and energy balance equations. To train the network, we perform numerical simulations on an ensemble of system parameters to obtain synthetic data. Even though the suggested approach provides results with the error of 3.96×10^{-4} which is in the same range as the result without physical regularisation, it is superior compared to conventional Artificial Neural Network (ANN) strategies because it ensures physical plausibility of the predictions, even in a highly dynamic and nonlinear problem. Consequently, the suggested PINN can be further developed for more complicated analysis of the TCES system.

Keywords: physics inspired neural network; physics-based regularisation; artificial neural network; nonlinear autoregressive network with exogenous input (NARX); thermochemical energy storage

1. Introduction

1.1. Thermochemical Energy Storage

Energy storage systems have become increasingly important in the shift towards renewable energy because of the fluctuation inherent to renewable energy generation [1,2]. Thermochemical Energy Storage (TCES) stores and releases energy in the form of heat as chemical potential of a storage material through a reversible endothermic/exothermic chemical reaction. TCES is favourable compared to sensible and latent heat storage [3–5] because it features a high energy density, low heat losses and the possibility to discharge the system at a relatively high and constant output temperature [6].

Numerous studies have been conducted on thermochemical energy storage in different materials, including calcium oxide [3,6], manganese oxide [7,8], barium oxide [9], zinc oxide [10], cobalt oxide/iron oxide [11], strontium bromide [12], calcium carbonate [13] and many more.

In general, the chemical processes occurring on the storage material can be classified into: redox of metal oxides, carbonation/decarbonation of carbonates and hydration/dehydration of hydroxides [14]. The choice of materials depends on many criteria, one of which is the application of the energy storage. For example, in integration with Concentrated Solar Power (CSP) plants, manganese oxide is not suitable because of its high reaction temperature [6,15]. Another important aspect to consider is the practicability of the process; for example, in a calcium carbonate system, CO_2 as the side effect of the reaction has to be liquefied and results in a high parasitic loss [6,14]. Additionally, there are many more criteria to consider, such as cyclability, reaction kinetics, energy density and, most importantly, safety issues. For comprehensive reviews of varying storage materials, we refer to [14–16].

Recently, experimental investigations have been conducted specifically for the calcium oxide (CaO)/calcium hydroxide ($Ca(OH)_2$) system. One experiment investigated the material parameters (such as heat capacity and density) and the reaction kinetics [17], another experiment focused on studying the operating range, efficiency and the cycling stability of the system [18], and there was also an experiment on the feasibility of integration with concentrated solar power plants [19]. All these experiments show that $CaO/Ca(OH)_2$ is a very promising candidate as TCES storage material. Furthermore, it is more attractive compared to other storage materials because it is nontoxic, relatively cheap and widely available [20,21]. The system stores the heat (is charged) during the dehydration of $Ca(OH)_2$ by injecting dry air with higher temperature. Charging results in an endothermic reaction along with the formation of H_2O vapour and lower temperature at the outlet. It releases the heat (is discharged) during the hydration of CaO. This is achieved by injecting air with higher humidity (H_2O content) and relatively lower temperature, resulting in an exothermic reaction (see Figure 1). Note that in this case, the hydration process occurs at lower temperature relative to the dehydration process, but both processes occur at high operating temperature [22]. The reversible reaction is written as:

$$Ca(OH)_2 + \Delta H \rightleftharpoons CaO + H_2O. \quad (1)$$

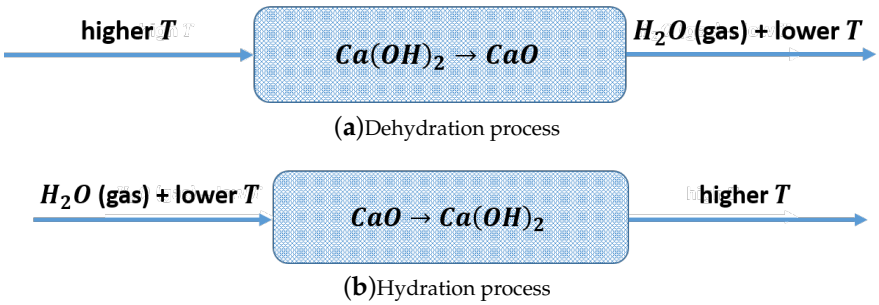


Figure 1. A simplified schematic of a Thermochemical Energy Storage (TCES) system with $CaO/Ca(OH)_2$ as the storage material during (**a**) dehydration and (**b**) hydration process.

A robust operational control of this system needs an accurate and real-time capable model to predict its state of charge and health. Similar models are operationally used, namely for batteries in mobile devices [23,24]. Accordingly, numerical TCES modelling studies were conducted to predict the system's behaviour [6,18,20,21,25]. However, the PDEs that describe the system are dynamic, highly nonlinear and strongly coupled, making the numerical simulation computationally expensive. This poses a significant hindrance on a more thorough and complex analysis of the TCES system. Estimation of the system's state of health, for example, requires a 2D or 3D model to study the effect of the structural change due to agglomeration [26]. With increasing spatial dimension, the computational time also increases strongly. Consequently, the system is not ready yet for commercial and industrial use until a faster and accurate model is developed. In this work, we consider using Artificial Neural Network (ANN) as a cheaper alternative to the expensive existing models.

1.2. Physics-Inspired Artificial Neural Networks

Artificial Neural Networks (ANNs) have been studied and applied intensively in the past few decades. They have become very popular alongside linear regression and other techniques such as Gaussian Process Regression (GPR) and Support Vector Machine (SVM) [27]. ANNs have advantages in terms of their flexibility and better applicability to model nonlinear problems compared to linear regression and GPR [28]. Additionally, it has better scalability to larger data compared to SVM [29]. However, a detailed performance comparison of ANN with other machine learning techniques is out of the scope of this paper.

ANNs have a wide range of applications, such as image and pattern recognition, language processing, regression problems and data-driven modelling [30]. In this paper, we focus on data-driven modelling, where an ANN is trained to predict the physical behaviour of a TCES system based on available data. ANNs have been used for data-driven modelling in different fields. In hydrology [31], ANNs have been successfully applied, for example to predict rainfall-runoff [32,33], groundwater levels [34] and groundwater contamination [35] . Moreover, ANNs have been used in energy system applications [36], for example to predict the performance of [37], reliability of [38] and design [39] renewable energy systems . All these examples show that ANNs have a potential to be a quick decision making tool which is useful for many engineering and management purposes.

In previous applications of ANNs in data-driven modelling, the ANN was treated as a black box [40,41] that learns only the mathematical relationship between the input and output. In such a process, the physical relationships and scientific findings that were previously used to build governing equations of the modelled systems are completely neglected. This issue is very troublesome and needs to be addressed because real data are noisy with measurement errors, and fitting the ANN to the noisy data without any physical constraint might lead to overfitting problems [30]. Additionally, in many cases, observation data is difficult and expensive to obtain, providing users with only a limited amount of data to train the ANN. Without any physical knowledge, ANNs perform poorly when trained with a low amount of data [42,43]. Furthermore, ANNs have a very poor interpretability [44,45], meaning that there might be different combinations of ANN elements (width, depth, activation functions and parameters) that fit the training data with similar likelihood, but not all of them are physically meaningful and robust. As a result, ANN predictions might be misleading.

Implementing physical knowledge to build the ANN structure and regularisation is a potential solution to solve this issue. By combining a black box model with a white box (fully physical) model, we obtain a grey box model. In such an approach, physical theories are used in combination with observed data to improve the model prediction and plausibility [46]. Moreover, the data will help to include complex physical processes that may not be captured in currently existing white box models. There are at least two motivations to do so: to obtain a reliable surrogate model for the physics-based model for the sake of speed in real-time environments and to address situations where the underlying physics of the system are incompletely understood, so that ANNs can build on, and later exceed, the current state of physical understanding.

Several works have been conducted to develop the so-called Physics Inspired Neural Networks (PINN). In general, PINNs can be grouped into two distinguishable motivations as mentioned above. The first one applies ANNs to infer the parameter values in the governing Ordinary Differential Equations (ODEs) or Partial Differential Equations (PDEs) as well as the constitutive relationships and the differential operators [43,47], assuming that the ODEs or PDEs perfectly describe the modelled system. The second one treats the system as a complex unit that is not sufficiently represented only with simplified equations. It trains the ANN based on observation data while constraining the ANN using physically-based regularisation [42,48,49]. We aim mostly at the second motivation and ask ourselves how much of the useful knowledge contained in the PDEs can be used to inform ANNs before proceeding to train them on observation data.

Despite the success of PINN implementations in this direction, there are still some open issues that need to be addressed: (I) There is no well-defined alignment between the structure of the governing

equations with the structure of the ANN. For example, most, if not all of the applications are for dynamic systems. Nevertheless, the structures of the ANNs applied do not resemble the dynamic behaviours of the systems and do not consider recurrency. (II) The focus in PINN development is more on getting high accuracy with limited amount of training data rather than improving the physical plausibility of the predictions. (III) The implementations of PINNs in previous works are mainly for relatively simple problems, and implementation to more complex problems (featuring multiple nonlinear coupled equations) has not been evident yet. In our current study, we address these three open issues.

1.3. Approach and Contributions

For dynamic and complex systems with coupled nonlinear processes such as the TCES system, we need an advanced approach to solve it using ANNs. Our approach implements physical knowledge of the system into building the ANN such as: (I) we use a Nonlinear Autoregressive Network with Exogeneous Inputs (NARX) structure. This is a form of Recurrent Neural Network (RNN), and we use deeper recurrence to account for the system's long-term time scales and nonlinear dynamics; (II) we train the network with recurrence structure to improve the long-term predictions in the dynamic system; and (III) we add physical regularisation terms in the objective function of training to enforce physical plausibility of the predictions.

NARX is suitable to model time series of sequential (time-dependent) observations $\boldsymbol{y}(t)$ [50,51], which are equispaced time series. There are several reasons why NARX is preferable to alternative ANN structures: the included feedback loop in NARX enables it to capture long-term dependencies [34] and the possibility to provide exogenous inputs improves the results compared to networks without them [52]. Thinking in terms of PDEs, the exogeneous inputs resemble time-dependent boundary conditions, and the feedback provides access to preceding time steps of the PDE solution. With deeper recurrence, even integro-differential equations can be resembled, which is important for hysteretic systems or for system descriptions on larger scales.

There are two different methods to train NARX, namely Series-Parallel (SP, also known as open-loop) and Parallel (P, also known as closed-loop). In SP training, each time step in the time series is used as an independent training example. This means the recurrency in the ANN structure is ignored, and the preceding data values from the time series are provided as feedback inputs instead of the predicted values. The feedback loop is closed only after completing the training to perform multistep ahead predictions [52,53]. The independency of the training examples makes the training much easier; however, the trained network performs much worse after closing the feedback loop [52].

Most, if not all studies conducted with NARX have used SP structure to train the network. In this paper, we argue that P training resembles the dynamic system better. The reason is that P training optimizes the ANN exactly for the later prediction purpose over longer time horizons: it accounts for error propagation over time and for the time-dependency of the predictions between time steps. As a downside, it requires more time to train the network in P mode. We are readily willing to accept this trade-off, because once trained, the network can still calculate its outputs in high speed [54]. We also propose to use a deeper recurrency to train the network by feeding back predictions of multiple preceding time steps. This accounts for the nonlinearity of the system and for possible higher-order memory effects in the system. In terms of PDE-governed systems, this corresponds to the time delay between system excitations at one system boundary and the system's reaction at a remote boundary.

Regularisation in training ANNs is useful to prevent overfitting. Here, as an addition to the commonly used L2 regularisation, simple regularisation terms are added that align with the physics. Several examples include, but are not limited to, monotonicity and non-negative values (examples in this case are volume and mole fractions) of the internal states. This follows the works presented in [46,55]. We suggest to use Bayesian Regularisation (BR) to optimally calculate the hyperparameters (normalising constants) of all terms in the loss function, unlike in previous works, where the hyperparameters were calibrated manually. Furthermore, regularisation terms with discretized balance

equations are also used. This regularisation is a way to feed the training with fundamental human knowledge previously used to build PDEs. It helps the network realise the extensive relationships between inputs (previous states and boundary conditions) and outputs (future states of the system) in complex problems and to prevent physically implausible predictions.

To test our ANN framework, a Monte Carlo ensemble of numerically simulated time series of system states is used, which we generate from random samples of uncertain system parameters. White noise is then added to the simulation results to emulate the actual noisy measurement data. Then, this ensemble (both parameters and time series) is used to train the network. We use synthetic data instead of experimental data because the former allow more exhaustive and controlled testing; this does not imply that our main purpose is only surrogate modelling. As optimization algorithm for training, the Levenberg–Marquardt (LM) algorithm [56–58] is implemented to obtain an optimum set of NARX parameters (which consist of the so-called weights and biases).

This paper is organised as follows: in Section 2 we introduce the governing equations used for numerical simulation of the TCES internal states, the alignment between the dynamic of $CaO/Ca(OH)_2$ and the NARX structure, as well as how we implement the physical knowledge into the regularisation. In Section 3, we discuss the results of our test, and Section 4 concludes the findings in the work.

2. Materials and Methods

2.1. Governing Equations

This study serves as an initial step towards enabling a more complex analysis of the TCES system, focusing on predictions of the system's dynamic internal states that change during the endothermic/exothermic reaction process. The analysis of the system's integration with the energy source is out of scope of this paper.

To set up the prediction model, we consider the $CaO/Ca(OH)_2$ TCES lab-scale reactor of 80 mm length along the flow direction as described in [20]. Assuming the system properties and parameters to be homogeneous, the simulation was conducted in 1D. The system was modelled as a nonisothermal single-phase multicomponent gas flow in porous media with chemical reaction acting as the source/sink terms and can be described using mole and energy balance equations. The inlet temperature and outlet pressure were fixed and defined with Dirichlet boundary conditions, and Neumann conditions were used to define the gas injection rates. The solid components forming the porous material are CaO and $Ca(OH)_2$, and the gases are H_2O and N_2. The latter serves as an inert component to regulate the amount of H_2O mole fraction in the injected gas. Full explanation in detail can be found in [20], and we offer a brief overview only in this section.

The mole balance equation was formulated for the solid component (subscript s) as:

$$\frac{\partial \rho_{n,s} \nu_s}{\partial t} = q_s, \tag{2}$$

where ρ_n denotes the molar density, ν the volume fraction, q the source/sink term, t the time and the subscript n refers to molar properties. Note that ν_s is the volume fraction of each solid component with regard to the full control volume, and therefore $\sum_s \nu_s = 1 - \phi$. In Equation (2), there is no effect of advection or diffusion (no fluxes) because the solid is assumed to be immobile. The change of solid component is solely caused by the chemical reaction through the reaction source/sink term q_s, assuming the solid is immobile. The change in the gas component (subscript g), however, is affected both by advective and diffusive mass transfer and by a source/sink term for the reactive component H_2O, as is defined in the mole balance equation:

$$\frac{\partial \rho_{n,g} x_g \phi}{\partial t} - \nabla . \left(\rho_{n,g} x_g \frac{K}{\mu} \nabla p + D \rho_{n,g} \nabla x_g \right) = q_g. \tag{3}$$

Here, x denotes the molar fraction, ϕ the porosity, K the absolute permeability of the porous medium, μ the gas viscosity, p the pressure and D the effective diffusion coefficient.

The energy balance equation was formulated assuming local thermal equilibrium. It accounts for internal energy change of both solid and gas phase, convective and conductive heat flux as well as source/sink term from the reaction. It was defined as:

$$\frac{\partial \rho_{m,g} u_g \phi}{\partial t} + \sum_s \frac{\partial (\nu_s \rho_{m,s} c_{p,s} T)}{\partial t} + \nabla . \left(\rho_{m,g} h_g \frac{K}{\mu} \nabla p \right) - \nabla . (\lambda_{eff} \nabla T) = q_e, \tag{4}$$

where ρ_m is the mass density, u_g is the gas specific internal energy, $c_{p,s}$ is the specific heat capacity of the solid material (CaO and $Ca(OH)_2$), T is temperature, h_g is gas specific enthalpy and λ_{eff} is the average thermal conductivity of both solid materials and gas components.

Reaction rates must be specified to determine the source/sink term for each equation. Based on [20,21], simple reaction kinetics were used, described as:

$$\hat{\rho}_{m,SR} = \begin{cases} -x_{H_2O} (\rho_{m,Ca(OH)_2} - \rho_{m,SR}) k_R^H \frac{T - T_{eq}}{T_{eq}}, & \text{if } T < T_{eq}, \\ -(\rho_{m,SR} - \rho_{m,CaO}) k_R^D \frac{T - T_{eq}}{T_{eq}}, & \text{if } T > T_{eq}, \end{cases} \tag{5}$$

where $\hat{\rho}_{m,SR}$ is the mass reaction rate, k_R^H and k_R^D are hydration and dehydration reaction constant, respectively and T_{eq} is the equilibrium temperature. Hydration process occurs when $T < T_{eq}$, which is also called the discharge process and is the exothermic part of the reaction; and dehydration process occurs when $T > T_{eq}$, also known as charge process and the endothermic part of the reaction. At the beginning of each reaction, the storage device is assumed to be in chemical equilibrium, corresponding to $\nu_{Ca(OH)_2} = 0$ and $\nu_{CaO} = 0$ for hydration and dehydration, respectively.

The relation between the reaction rate and the source/sink terms for the mole balance equations were defined as:

$$q_{H_2O} = q_{CaO} = -q_{Ca(OH)_2} = \frac{\hat{\rho}_{n,SR}}{1 - \phi}, \tag{6}$$

with $\hat{\rho}_{n,SR}$ the molar reaction rate (obtained from $\hat{\rho}_{m,SR}$ using the molar mass of each respective component). The energy balance source/sink term q_e was calculated accounting for the reaction enthalpy ΔH and the volume expansion work [59] according to:

$$q_e = -\hat{\rho}_{n,SR} \left(\Delta H - \frac{\phi}{1 - \phi} \frac{p}{\rho_{n,g}} \right). \tag{7}$$

Note that a negative sign is necessary to calculate q_e, so that its value is in proportion to $q_{Ca(OH)_2}$, and in reverse to q_{H_2O} and q_{CaO}. This negative sign can be explained by the fact that to form $Ca(OH)_2$ from CaO and H_2O in the hydration process, energy is released into the system. Correspondingly, a decrease in the molar amount of CaO and H_2O (and an increase in the molar amount of $Ca(OH)_2$) results in a positive source term. The opposite holds for the dehydration process.

2.2. Input and Output Variables

The numerical model used in this work was developed using DuMu^x (Distributed and Unified Numerics Environment for Multi-{Phase, Component, Scale, Physics, ...} [60]). As input to the simulator, we need the material parameters such as CaO density (ρ_{CaO}), $Ca(OH)_2$ density ($\rho_{Ca(OH)_2}$), CaO specific heat capacity ($c_{p,CaO}$), $Ca(OH)_2$ specific heat capacity ($c_{p,Ca(OH)_2}$), CaO thermal conductivity (λ_{CaO}) and $Ca(OH)_2$ thermal conductivity ($\lambda_{Ca(OH)_2}$); porous medium parameters such as absolute permeability (K) and porosity (ϕ); reaction kinetics parameters such as reaction rate constant (k_r) and specific reaction enthalpy (ΔH); and initial and boundary conditions such as N_2 molar inflow rate ($\dot{n}_{N_2,in}$), H_2O molar inflow rate ($\dot{n}_{H_2O,in}$), initial pressure (p_{init}), outlet pressure (p_{out}), initial temperature (T_{init}), inlet temperature (T_{in}) and initial H_2O mole fraction ($x_{H_2O,init}$).

In the TCES system application, one of the main goals is to estimate the state of charge of the device that is implied in the CaO volume fraction ν_{CaO}. The device in fully charged condition corresponds to $\nu_{CaO} = 1$ and vice versa. We are also interested in the output variables p, T and x_{g,H_2O} (H_2O mole fraction). The behaviour of these variables, especially p, is very nonlinear. Therefore, it is interesting to see the prediction of the ANN for these nonlinear variables. Additionally, these variables are also important to assist in the system understanding. Therefore, our main output variables of interest were defined as in the following vector $\boldsymbol{y}$ as a function of time t:

$$\boldsymbol{y}(t) = \begin{bmatrix} p(t) \\ T(t) \\ \nu_{CaO}(t) \\ x_{g,H_2O}(t) \end{bmatrix}. \tag{8}$$

All input-output data samples are available as supplementary materials on https://doi.org/10.18419/darus-633.

2.3. Aligning the ANN Structure with Physical Knowledge of the System

ANN representation via NARX has two different training architectures, namely Series-Parallel (SP) and Parallel (P) structure. The network output $\hat{\boldsymbol{y}}_{SP}(t+1)$ of the SP structure is a function of the observed target values of previous time steps $\boldsymbol{y}(t)$ up to a feedback delay d_y and of the so-called exogenous inputs $\boldsymbol{u}$:

$$\hat{\boldsymbol{y}}_{SP}(t+1) = f\Big(\boldsymbol{y}(t), \boldsymbol{y}(t-1), \ldots, \boldsymbol{y}(t-d_y), \boldsymbol{u}\Big). \tag{9}$$

In this work, $\boldsymbol{u}$ was assumed to be constant over time, meaning there is no disturbance signal throughout the whole simulation period.

In P structure, the difference lies in the fed-back values. Here, the network outputs of the P structure $\hat{\boldsymbol{y}}_P(t)$ are fed-back instead of the original given data $\boldsymbol{y}(t)$:

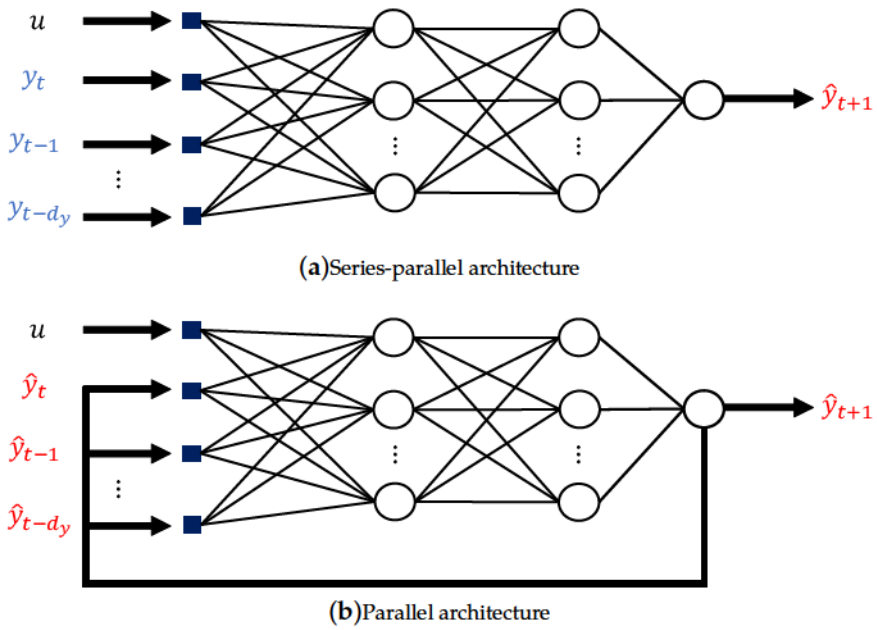
$$\hat{\boldsymbol{y}}_P(t+1) = f\Big(\hat{\boldsymbol{y}}(t), \hat{\boldsymbol{y}}(t-1), \ldots, \hat{\boldsymbol{y}}(t-d_y), \boldsymbol{u}\Big). \tag{10}$$

Note that, in terms of notation, the difference only lies in the hats above the fed-back values. Apparently, the P-structure in NARX resembles an explicit time-discrete differential equation (ODE or PDE) in a simplistic case, for example using the Adams–Bashforth discretization scheme [61,62] which can be described as:

$$\hat{y}(t+1) \approx \hat{y}(t) + \Delta t \cdot g(u, \hat{y}(t), \hat{y}(t-1), \ldots, \hat{y}(t-d_y)). \tag{11}$$

where $\hat{y}(t+1)$ is an explicit function of $\hat{y}(t) \ldots \hat{y}(t-d_y)$. In Equation (10), the NARX function $f\Big(\hat{\boldsymbol{y}}(t), \hat{\boldsymbol{y}}(t-1), \ldots, \hat{\boldsymbol{y}}(t-d_y), \boldsymbol{u}\Big)$ can be seen as an approximation of $\hat{y}(t) + \Delta t \cdot g(u, \hat{y}(t), \hat{y}(t-1), \ldots, \hat{y}(t-d_y))$ in Equation (11). Based on this reason, we propose to train using P-structure for solving dynamic problems whenever possible. Additionally, training in P-structure helps the network to learn that there is dependency between predicted values at different time steps. While both architectures were considered for NARX training, only P architecture was used for testing, as for longer-term forecasting, real data of previous time steps are not available [52]. For better understanding of the difference between P and SP, Figure 2 illustrates both architectures.

Feedback delay is also an important property, because Equations (2)–(4) are not elliptic, and hence there will be a time delay for the effect of input change to change the output. Because of this memory effect and its nonlinearity, we propose to use a deeper recurrence in NARX to enable the network to learn the system's latency. In this work, feedback delay values d_y ranging from 1 to 5 were tested to get the optimum value.



(a) Series-parallel architecture

(b) Parallel architecture

Figure 2. Difference between (**a**) SP and (**b**) P architecture. Here, $y_t ... y_{t-d_y}$ (in blue) are the given data, while $\hat{y}_t ... \hat{y}_{t-d_y}$ (in red) are the ANN predictions.

Additional to an appropriate ODE-like structure, the hyperbolic tangent (tanh) function was chosen as activation function within the neurons of NARX. Tanh is a nonlinear activation function (named tansig function in MATLAB [63]). Aside from the nonlinearity, this choice was driven by the assumption that all the input parameters and the targets depend on each other via smooth functions (differentiable). This knowledge results, among others, from the presence of a diffusion term in all relevant transport equations, and from the absence of shock waves in the solutions to Equations (2)–(4). Hence, for each hidden layer l as shown in Figure 2, the layer output $\boldsymbol{a}^{[l]}$ is computed in the feedforward procedure described in:

$$\boldsymbol{a}^{[l]} = \tanh(\boldsymbol{w}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}). \tag{12}$$

For the output layer L, a linear function is assigned for scaling, leading to:

$$\hat{\boldsymbol{y}}^{[L]} = \boldsymbol{w}^{[L]}\boldsymbol{a}^{[L-1]} + \boldsymbol{b}^{[L]}. \tag{13}$$

2.4. Physical Constraints in the Training Objective Function

As loss function for training, the most commonly used performance measure is a Mean Squared Error (MSE). Equation (14) shows the MSE for n training datasets (here, we use the subscript D for "data" to label the error term E_D in the loss function as the data-related term):

$$E_D = \frac{1}{n.n_t}\sum_{i=1}^{n}\sum_{t=1}^{n_t}(y_{i,t} - \hat{y}_{i,t})^2, \tag{14}$$

where $i = 1 \ldots n$ indicates a specific sample in the training dataset and $t = 1 \ldots n_t$ indicates a specific time step. However, using MSE alone in the loss function is not enough most of the time. The optimization problem to be solved in training is typically an ill-posed problem in many instances [30]. Thus, regularisation is required to prevent overfitting.

In this work, the L2 regularisation method was used to increase the generalisation capability of the ANN [64]. L2 regularisation is also known as weight decay or ridge regression [65]. The goal of L2 regularisation is to force the network to have small parameter values (choosing the simpler network

over the more complex one). This effectively adds a soft constraint to the loss function to prevent the network from blindly fitting the possible noise in training datasets:

$$E_\theta = \frac{1}{N}\sum_{j=1}^{N}\theta_j^2, \tag{15}$$

where N is the total number of network parameters (weights and biases), and $\theta \in \mathbb{R}^N$ are the network parameters.

To improve the network prediction and the physical plausibility even more, known physical laws were inserted as part of the network regularisation:

$$E_{phy,k} = \frac{1}{n.n_t}\sum_{i=1}^{n}\sum_{t=1}^{n_t}(e_{phy,i,t,k})^2, \tag{16}$$

where the subscript k identifies a specific physical law, for example a mole balance equation, and $e_{phy,i,t,k}$ is the physical error listed in Table 1. For example, the term $e_{phy,i,t,1}$ corresponds to the mole balance equation for dehydration/hydration. The mole balance equation used for this regularisation is the H_2O mole balance, because it has the most complete storage, flux and source/sink term (the solid components are assumed to be immobile, and N_2 is inert). The mole balance error can be written as:

$$e_{MB}(i,t) = n_{H_2O,out}(i,t) - n_{H_2O,in} + \Delta n_{H_2O,sto}(i,t) - \Delta n_{H_2O,q}(i,t), \tag{17}$$

where n_{H_2O} is the molar amount of H_2O, the subscript out, in, sto and q denote outflow, inflow, storage and source/sink term, respectively. The mole balance error was used as a contraint $e_{phy,i,t,1}$ and is equal to 0 if the mole balance is fulfilled. Putting this equation as a regularisation term penalises the network if the mole balance is not satisfied. Similarly, the corresponding energy balance equation also has to be fulfilled:

$$e_{EB}(i,t) = Q_{out}(i,t) - Q_{in} + \Delta Q_{sto}(i,t) - \Delta Q_q(i,t), \tag{18}$$

where Q is the energy in the system. It was used as a regularisation in $e_{phy,i,t,2}$. A more detailed derivation of the mole balance error $e_{phy,i,t,1}$ and the energy balance error $e_{phy,i,t,2}$ can be found in Appendix B.

Table 1. Physical constraints in training: loss term used in Equation (16).

k	**Equation:** $e_{phy,i,t,k} = ...$	
	Dehydration	**Hydration**
1	$e_{MB}(i,t)$	
2	$e_{EB}(i,t)$	
3	$ReLU(-\hat{v}_{CaO}(i,t))$	
4	$ReLU(-\hat{x}_{H_2O}(i,t))$	
5	$ReLU(\phi + \hat{v}_{CaO}(i,t) - 1)$	
6	$ReLU(\hat{p}(i,t-1) - \hat{p}(i,t))$	$ReLU(\hat{p}(i,t) - \hat{p}(i,t-1))$
7	$ReLU(\hat{T}(i,t-1) - \hat{T}(i,t))$	$ReLU(\hat{T}(i,t) - \hat{T}(i,t-1))$
8	$ReLU(\hat{v}_{CaO}(i,t-1) - \hat{v}_{CaO}(i,t))$	$ReLU(\hat{v}_{CaO}(i,t) - \hat{v}_{CaO}(i,t-1))$
9	$ReLU(\hat{T}(i,t) - T_{in})$	$ReLU(T_{in} - \hat{T}(i,t))$

Further relations of the form $\mathcal{F}(\hat{y}) \leq 0$ (monotonicity and non-negative values) were implemented using the Rectified Linear Units (ReLU) function, so that the physical error was then calculated with $e_{phy,i,t,k} = ReLU(\mathcal{F}(\hat{y}))$ [46]. The ReLU function returns 0 as output value for negative arguments

and linearly increases for positive arguments. Hence, it punishes positive values in proportion to their magnitudes. Examples of these ReLU constraints are $e_{phy,i,t,3}$ through $e_{phy,i,t,13}$. They define non-negativity and monotonicity of the predicted target variables. For both dehydration and hydration process, negative fractional values $\hat{v}_{CaO}$ and $\hat{x}_{H_2O}$ are physically and mathematically impossible. Therefore, in $e_{phy,i,t,3}$ and $e_{phy,i,t,4}$, the network is punished for predicting negative values for these targets. Additionally, for both processes, $e_{phy,i,t,5}$ provides an additional constraint for $\hat{v}_{CaO}$, to limit the amount of CaO volume in relation with the porosity ($\hat{v}_{CaO} \leq 1 - \phi$). All these monotonicity assumptions originated from the fact that the system's material parameters are considered to be constant throughout operation of the system. Therefore, the system's behaviour should be monotonic and bounded in the specified aspects. Specific for the dehydration process, $\hat{p}$, $\hat{T}$ and $\hat{v}_{CaO}$ are expected to not decrease throughout the simulation. This results in the corresponding monotonicity constraints $e_{phy,i,t,6}$ to $e_{phy,i,t,8}$. The system temperature must also be lower or equal to the injected temperature as constrained in $e_{phy,i,t,9}$, because the injected temperature is higher than the initial temperature. Specifically for the hydration process, the monotonicity constraints $e_{phy,i,t,6}$ to $e_{phy,i,t,9}$ for the dehydration process are reversed, because the hydration process is the reverse of the dehydration process.

2.5. Obtaining Optimum Network Parameters

The complete loss function defined in Section 2.4 including MSE and all the regularisation term is written as:

$$L(\theta) = \alpha E_\theta + \beta E_D + \sum_k \lambda_k E_{phy,k}. \tag{19}$$

Here, α and β are normalising constants of E_θ and E_D, respectively; and λ_k is a normalising constant for each physical regularisation k. All error and regularisation terms, therefore, are evaluated in a normalised metric. These normalising constants, also known as the hyperparameters, determine the importance given to each term. For example, a high β means that it is more important for the network to fit the training datasets than to generalise better. In many cases, the hyperparameters are determined manually from trial-and-error. In this work, Bayesian Regularisation was adopted to calculate them overall using a maximum likelihood approach to minimise $L(\theta)$ [66,67]. Bayesian Regularisation reduces the subjectivity arising from manual choice of hyperparameters. First, all the hyperparameters α, β and λ_k along with the network parameters θ were initialised. The hyperparameters were initialised by setting $\beta = 1$, $\alpha = 0$ [68] and also $\lambda_k = 0$, while the network parameters were initialised using the Nguyen-Widrow method [69,70]. The Nguyen-Widrow method initialises the network parameters so that each neuron contributes to a certain interval of the whole output range (added with some random values).

The complete derivation for updating α and β can be found in [66,67]. Here, we only give the calculation of λ_k. After each iteration, they are updated according to the following relation:

$$\lambda_k := \frac{N - \lambda_k Trace(\boldsymbol{H}^{-1} \boldsymbol{J}_{phy,k}^T \boldsymbol{J}_{phy,k})}{2E_{phy,k}}, \tag{20}$$

where $\boldsymbol{J}_{phy,k}$ is the Jacobian of physical error $E_{phy,k}$ with respect to the network parameters θ. The approximate Hessian matrix $\boldsymbol{H}$ of the overall loss function $L(\theta)$ was defined as follows:

$$\boldsymbol{H} = \alpha \boldsymbol{I} + \beta \boldsymbol{J}_D^T \boldsymbol{J}_D + \sum_k \lambda_k \boldsymbol{J}_{phy,k}^T \boldsymbol{J}_{phy,k}, \tag{21}$$

where $\boldsymbol{J}_D$ is the Jacobian of the MSE (E_D) with respect to the network parameters and $\boldsymbol{I}$ is the $N \times N$ identity matrix (N is the number of network parameters θ).

The network parameters are also updated after each iteration according to the Levenberg–Marquardt algorithm:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - (\boldsymbol{H} + \boldsymbol{I}\mu)^{-1}(\alpha\boldsymbol{\theta} + \beta\boldsymbol{J}_D E_D + \sum_k \lambda_k \boldsymbol{J}_{phy,k} E_{phy,k}), \tag{22}$$

where $\mu > 0$ is the algorithm's damping parameter. Its value is increased when an iteration step is not successful, and is decreased otherwise. The Levenberg–Marquardt algorithm was chosen because of its faster convergence rate compared to Steepest Decent and higher stability relative to the Gauss–Newton algorithm [58]. In absence of our physical regularisation terms and with fixed α, β, the procedure would simplify to a plain (nonlinear) least squares training, which would be the standard approach for training ANNs. Values of the trained network parameters and the normalising constants at the end of the training are given as supplementary materials on https://doi.org/10.18419/darus-634.

3. Results and Discussion

Our hypothesis is that applying physical knowledge of the modelled system into the construction of ANNs would lead to an improved physical plausibility of the prediction results. In this section, the prediction of the TCES system using ANNs is assessed and three relevant aspects that support our hypothesis are discussed: (I) the effect of feedback delay on the prediction result to account for the system's nonlinearity and long-term memory effect (Section 3.1), (II) the comparison between training in SP and P architecture (Section 3.2) and (III) the improved physical plausibility from using physical regularisation (Section 3.3). The results are illustrated only for the dehydration process, because the hydration provides very similar results.

The complete workflow of the ANN application is shown in Figure 3. In general, the workflow can be divided into: training, validation and testing of the ANN. To train the ANN, first an ensemble of exogeneous input $\boldsymbol{u}$ was generated based on selected probability distributions. These distributions are based on different values used in literature [6,17–21]. The complete list of exogeneous inputs $\boldsymbol{u}$ with their corresponding distributions is listed in Appendix A. This ensemble was then plugged into the numerical model in DuMu^x and was simulated until $t = 5000$ s to obtain an ensemble of target data $\boldsymbol{y}(t)$. The governing equations are provided in Section 2.1. White noise was then added to these targets by generating normally distributed random numbers with zero mean and a standard deviation of 0.05 times the target values. Lastly, both exogenous inputs and targets were normalised to the range $[-1, 1]$ to help the stability of the training [71]. Then, we set up the NARX ANN as described in Section 2. The training was then conducted using the built-in functions for NARX in the MATLAB Neural Network Toolbox [63], in which the loss function calculation was modified based on the equations provided in Section 2.5. It was conducted in batch mode both for dehydration and hydration process with a total of 100 training datasets.

Without physical regularisation, we obtain the lowest MSE value when the NARX is trained using 1000 training datasets, as shown in Figure 4. However, it is interesting to see how the physical knowledge can further improve the performance of NARX with limited training data. Therefore, we conducted the training in batch mode both for dehydration and hydration process with a total of only 100 training datasets.

For conciseness, the choice of the number of hidden layers, number of nodes per hidden layer and choice of activation function is not discussed because there is no existing uniform and systematic method to calculate the appropriate or the best combination [72]. Based on trial and error, we found that for this specific problem, 2 hidden layers with 15 and 8 nodes at each layer gives reasonable results. An example of ANN prediction using this configuration is provided in Appendix C. The stopping criteria are the dampening factor $\mu > 10^{10}$ (see Equation (22)) or the loss function gradient $g = \frac{\partial L(\theta)}{\partial \theta} < 10^{-7}$, both of which are the default values proposed in the toolbox. Additionally, a maximum epochs is set for the training. Since training error converges mostly before 500 epochs, this limit is sufficient.

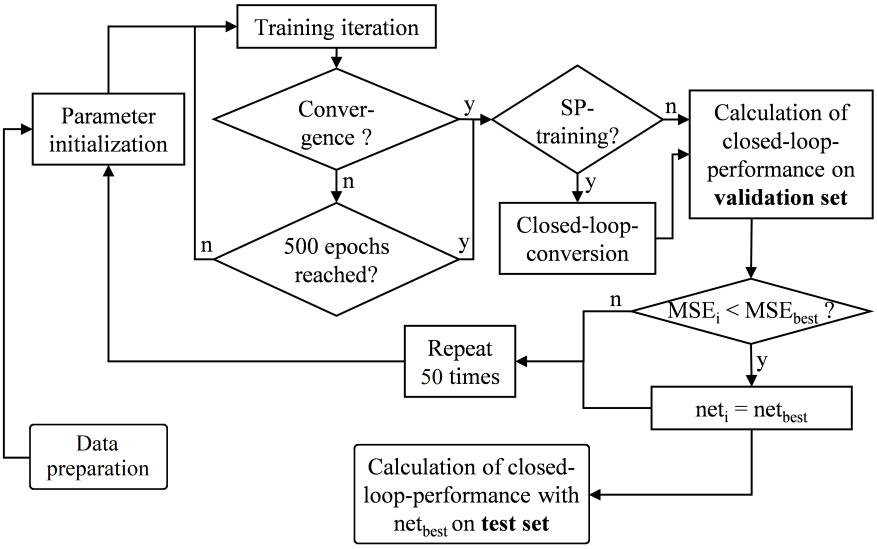


Figure 3. Flowchart of training, validation and test of the ANN.

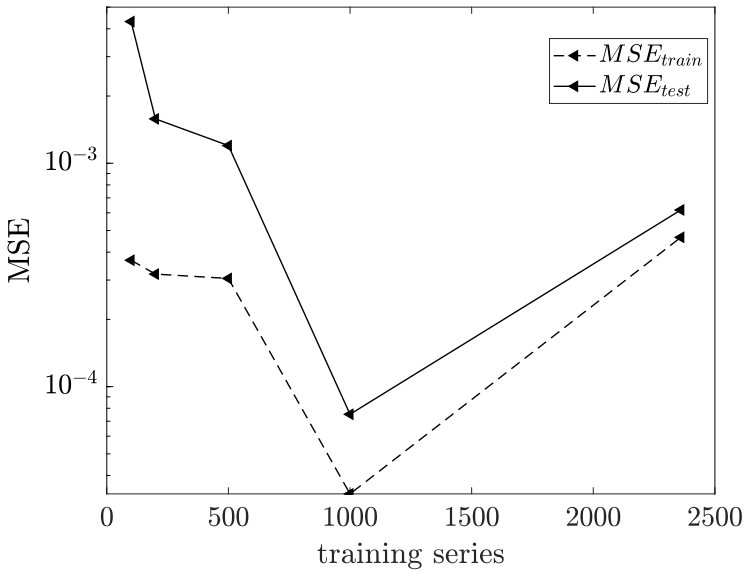


Figure 4. Comparison of MSE using different number of training datasets.

Different initialization values often lead to different network response as the algorithm might fail to always locate the global minimum [65]. Therefore, the network was retrained with 50 different initialisations. This number was set to find a compromise between a reliable result and a reasonable computational time. After each training, the network was validated on 20 validation datasets, and the trained network with the lowest MSE (E_D evaluated against validation data) was selected. Finally, the network was tested with data contained neither in the training nor in the validation set on a test set with 800 time series.

3.1. Influence of Feedback Delay

Figure 5 shows the influence of feedback delay on the MSE evaluated on both the training (dashed lines) and test datasets (solid lines) for networks trained using P structure. As shown in Figure 5, for $d_y > 1$ the test MSE is lower than the training MSE. This is generally because the network was trained using additional white noise—producing more errors in the training, while the test datasets were smooth for reference.

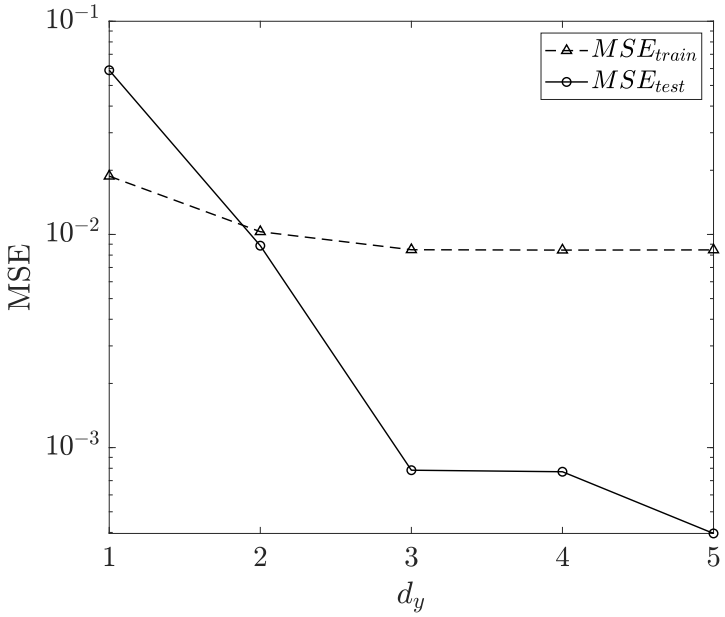


Figure 5. Feedback delay variation.

In Figure 5, while the training error seems to remain constant at $d_y > 2$, the test error keeps decreasing with increasing d_y. This clearly illustrates that including more depth in the recurrence improves the generalisation capability and therefore improves the ANN prediction. As the best MSE_{test} was obtained with $d_y = 5$, this value will be used from here onwards. From Figure 5 we can also analyse that a time delay of at least 3 previous time steps is useful to train the network. Moreover, we do not see the value of using $d_y > 5$, as judging by the MSE trend in Figure 5, there is no significant improvement expected.

3.2. SP Versus P Training Structure

Figure 6 compares the training time of P compared to SP structure. Moreover, plotting the gradient and performance as function of training epochs allows us to analyse the difference between both training characteristics. As expected and shown in Figure 6 (dashed lines), both SP and P structure training time increases nearly linearly with the number of epochs (iterations). However, the slope is steeper for P training which is caused by higher computational cost using Backpropagation Through Time (BPTT).

In Recurrent ANNs, BPTT is used to calculate the derivative of the loss function instead of the normal backpropagation method. BPTT is technically the same as the normal backpropagation method but with the RNN unfolded through time being the main difference [30]. The gradient is then backpropagated through this unfolded network. Unfolding the recurrent network increases its size, and therefore the optimization problem becomes computationally more expensive and more difficult to solve.

After every epoch in P training, the output values change, and consequently the feedback values also change. The constantly changing feedback values cause additional changes of the gradient values along the iteration (dotted lines in Figure 6). This makes the training a more nonlinear problem. Correspondingly, the training performance (smaller MSE) increases much slower. In SP training, the gradient strongly decreases during the first 20 epochs, showing that the SP training is more computationally stable. However, the MSE (solid lines in Figure 6) was evaluated during training for the structure the network was trained with, meaning that the training performance does not consider the closed-loop conversion error for SP training. For that reason, the MSE values shown in Figure 6

seem to be better for SP training. Regardless of this difference, both training procedures converge with strictly monotonic decay of their MSE.

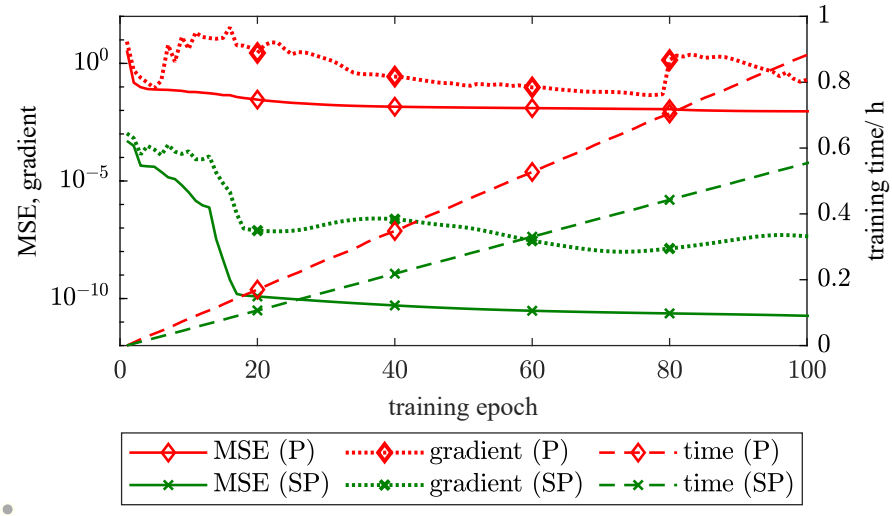


Figure 6. Training time, gradient and performance for P (Parallel) and SP (Series-Parallel) structure.

Next, the prediction performances of both training architectures are compared. In Figure 7, the results of the SP-trained NARX (dashed lines) are shown compared to the target values obtained from the simulation (solid lines) as reference solutions. Here, all target variables are calculated with differing inlet temperature. After a few time steps with relatively precise forecasts, the NARX predictions for T, p and ν_{CaO} diverge from real values and are highly fluctuating over time, which is nonphysical. Note that, in Figure 7, the results are shown only up to time step 100 instead of 1000. This is because the NARX prediction results after time step 100 have even higher fluctuations as the error propagates, hence making the comparison unclear visually. The forecast for x_{g,H_2O} is reasonably accurate for $t < 100$ but more erroneous for longer forecast periods. The forecast error is caused by the closed-loop-instability, meaning the inaccuracy caused by converting the network structure from SP to P. In other words, training using SP structure gives increasingly erroneous results with increasing time horizon. On the contrary, training with P structure provides clearly more accurate forecasts compared with SP, as shown in Figure 8. The NARX predictions (dotted lines) correspond really well to the reference targets (blue solid lines) throughout the whole simulation time for 1000 time steps, with inlet temperatures covering the whole range of its input distribution.

The comparison of P and SP structure shows that, while training in P structure seems to be more unstable, it provides significantly better long-term predictions because it trains the network to realise the time-dependency of output variables in a dynamic system. As shown by Equation (11), NARX resembles an explicitly discretized ODE, which is known to be conditionally stable. In cases where the discrete ODE is unstable, the error grows exponentially through time [73]. By training the network using P structure, the same structure is used for both training and testing, hence minimising the error propagation. The higher computing time needed to train in P structure (in comparison with training in SP structure) should not be a problem because the training only needs to be conducted once. Once the optimum network parameters are obtained, NARX can give reasonably accurate predictions with fast computational time. In fact, almost all studies in the area of surrogate models are willing to accept high computational costs during training (called offline costs) if the accuracy and speed for prediction (online) are good [54,74,75].

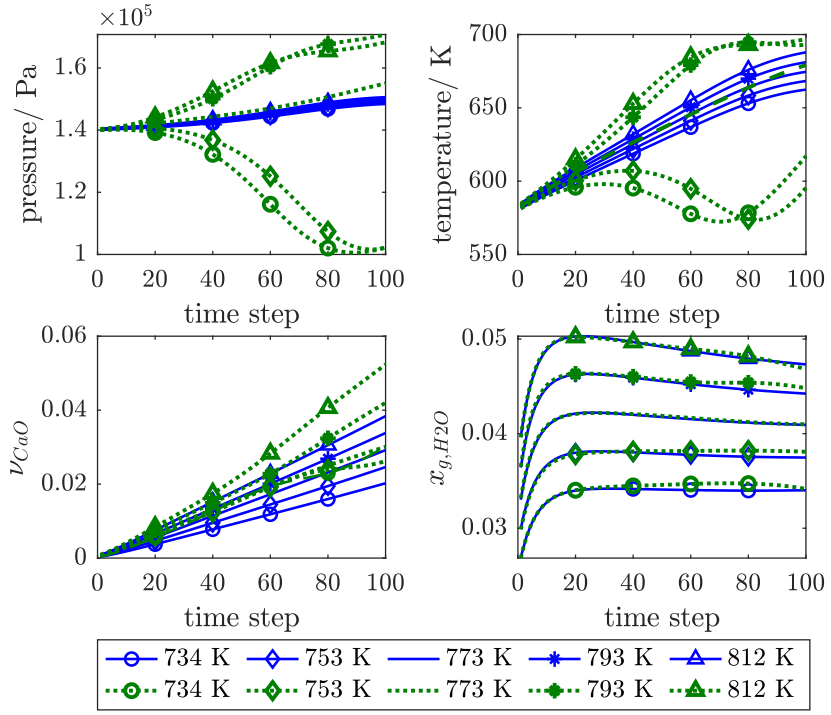


Figure 7. Forecasts with SP-trained NARX for various inlet temperatures.

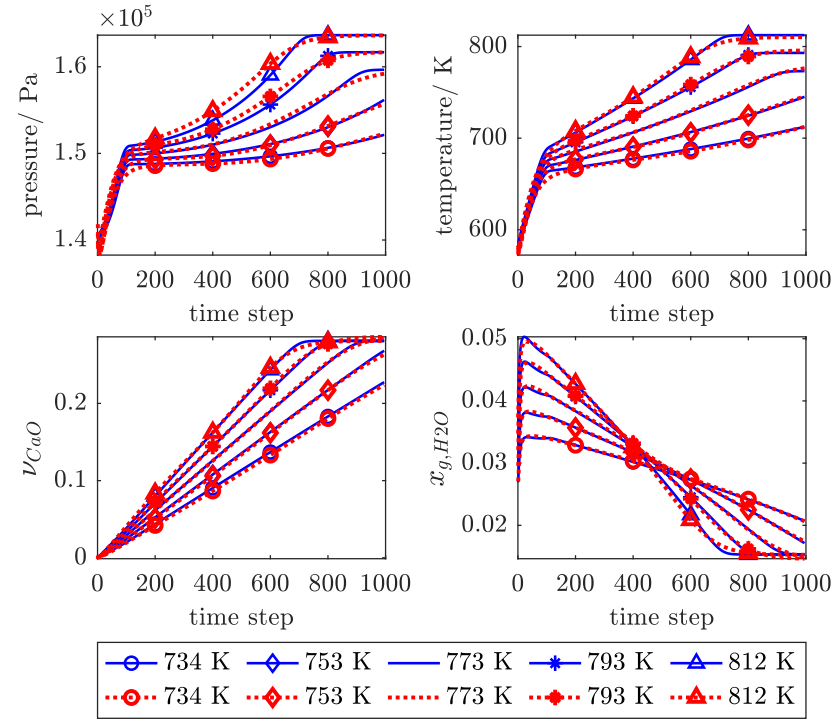


Figure 8. Forecasts with P-trained NARX for various inlet temperatures.

3.3. Physical Regularisation Improves Plausibility

To test the benefit of physically-based regularisation, the ANN performance is compared between training the network using:

- only MSE as the loss function ("MSE"), that is $L(\theta) = E_D$,
- MSE combined with only L2 ("MSE + L2"), that is $L(\theta) = \alpha E_\theta + \beta E_D$,

- MSE combined with only physical regularisation ("MSE + PHY"), that is $L(\theta) = \beta E_D + \sum_k \lambda_k E_{phy,k}$, and
- MSE combined with both L2 and physical regularisation ("MSE + L2 + PHY"), that is $L(\theta) = \alpha E_\theta + \beta E_D + \sum_k \lambda_k E_{phy,k}$.

To make a fair comparison, all networks were trained with the same set of initial network parameters. In this test, we used only P training and feedback delay $d_y = 5$, because they clearly showed preferable performance in the previous sections. The comparison is summarised in Table 2.

Table 2. MSE for different regularisation methods

Loss Function	MSE_{train}	MSE_{test}
MSE	8.45×10^{-3}	2.81×10^{-3}
MSE + L2	9.01×10^{-3}	3.96×10^{-4}
MSE + PHY	8.68×10^{-3}	3.83×10^{-3}
MSE + L2 + PHY	8.43×10^{-3}	3.96×10^{-4}

While the MSE on training data is in a comparable range for all loss functions, differences in MSE_{test} are observed. L2 regularisation helps to reduce overfitting of training data, resulting in a lower test error in "MSE + L2" compared to "MSE". At first glance, the additional physical regularisation does not seem to further improve the results. MSE_{test} of "MSE + PHY" is slightly worse compared to "MSE", and MSE_{test} of "MSE + L2 + PHY" is in the same order with "MSE + L2" because another constraint is added in the objective function, while the performance is measured only based on MSE. Moreover, using only the physical (MSE + PHY) instead of only L2 regularisation (MSE + L2) leads to a test performance decrease.

Even though the performance for both "MSE" and "MSE + L2" are better than "MSE + PHY", they both fail to produce physically plausible predictions in several test datasets (outliers) as shown in Figures 9 and 10 (the label "Reference" for the blue line refers to the synthetic test data obtained from the physical model), the clearest one being negative fraction values of CaO and H_2O. One important aspect that needs to be considered is that the ANN was trained using only 100 training datasets, compared to almost 500 parameters that exist inside the network. This made the optimisation problem an ill-posed one, leading to clear overfitting in the network with "MSE" and "MSE + L2". Physical regularisation tackles this problem even for relatively sparse training data, which is valuable once experiments are costly, and therefore, not much data are often available to train the network.

Even though it produces the worst overall test performance, physical regularisation alone (MSE + PHY) is able to produce physically plausible results despite no application of L2 regularisation, see Figure 11. The figure illustrates the worst prediction result of all test sets obtained using "MSE + PHY". Even in its worst prediction with high error, the network is much more stable. With the addition of L2 regularisation in the "MSE + L2 + PHY" scheme, the prediction error (MSE) is further reduced so that it lies within the same range as "MSE + L2". The major difference here is illustrated in Figure 12, where the worst prediction result produced by "MSE + L2 + PHY" is far more physically plausible, shown by the absence of unstable fluctuations as well as the relatively higher accuracy.

We trained the ANN using numerical simulation results which indirectly imbues the physics from the formulated governing equations into the ANN. When the ANN was not trained using numerical simulation results but with real observation data (which could follow more complex, scientifically unexplored equations), physical regularisation helps to constrain the ANN training at least to fundamental, confirmed laws and prevent unnecessary overfitting to the irregular and noisy observation data. As such, implementation of the method we present will be even more beneficial for applications with real observation data.

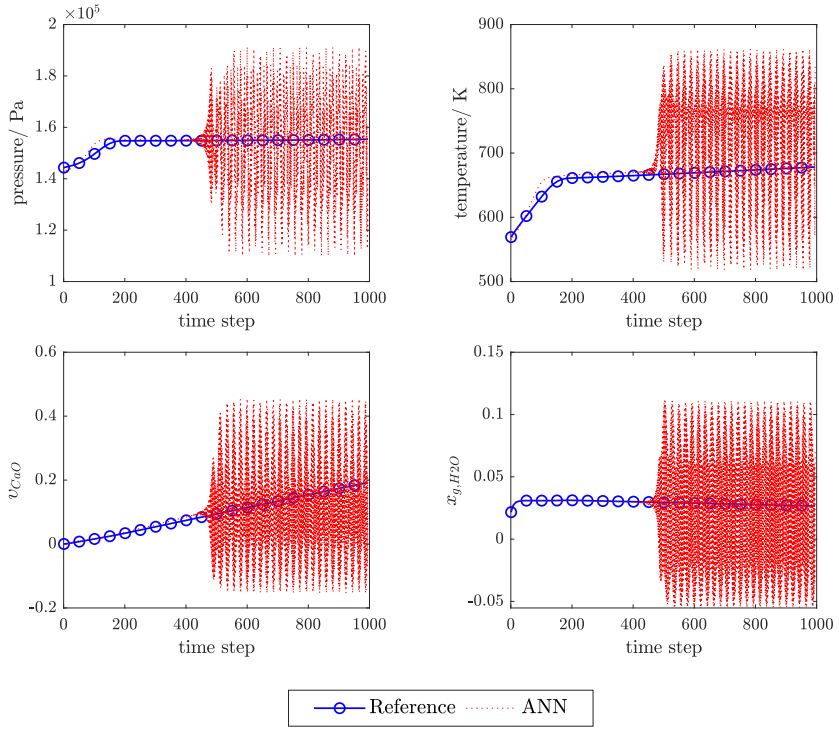


Figure 9. Worst prediction sample (red) obtained with "Mean Squared Error—MSE" regularisation method and reference solution obtained from the physical model (blue).

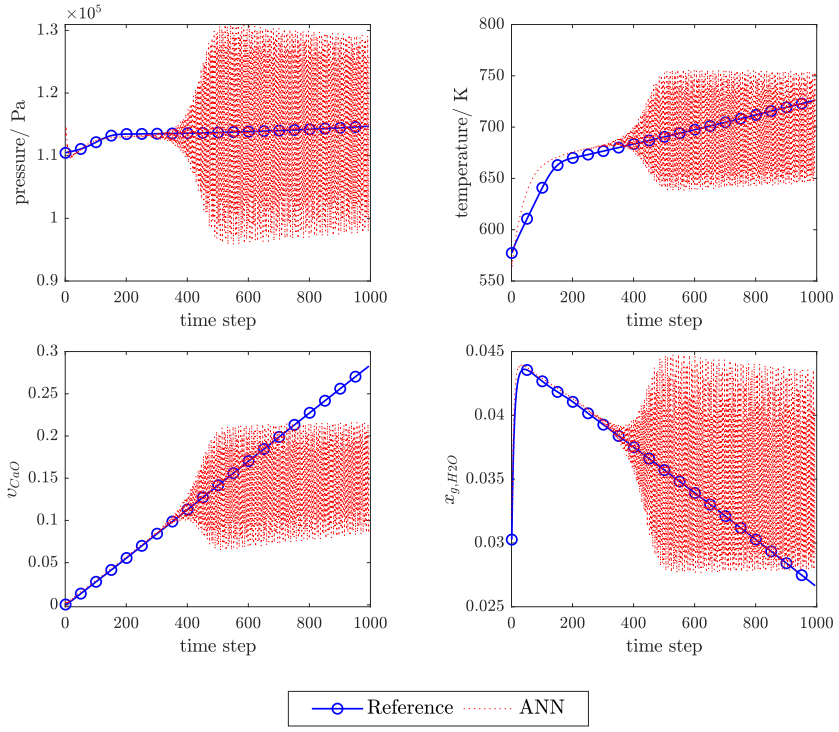


Figure 10. Worst prediction sample (red) obtained with "MSE + L2" regularisation method and reference solution obtained from the physical model (blue).

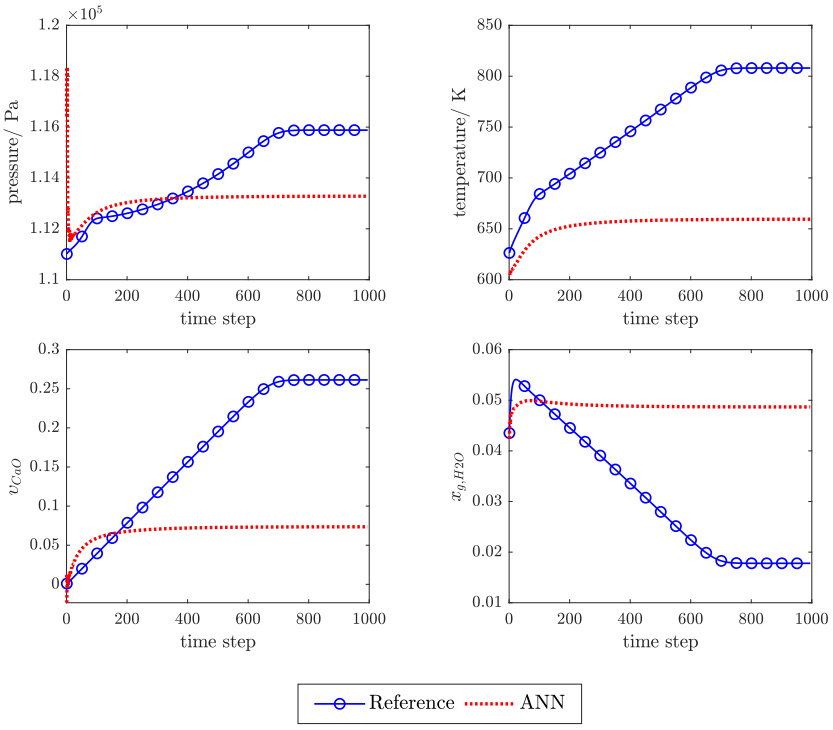


Figure 11. Worst prediction sample (red) obtained with "MSE + PHY" regularisation method and reference solution obtained from the physical model (blue).

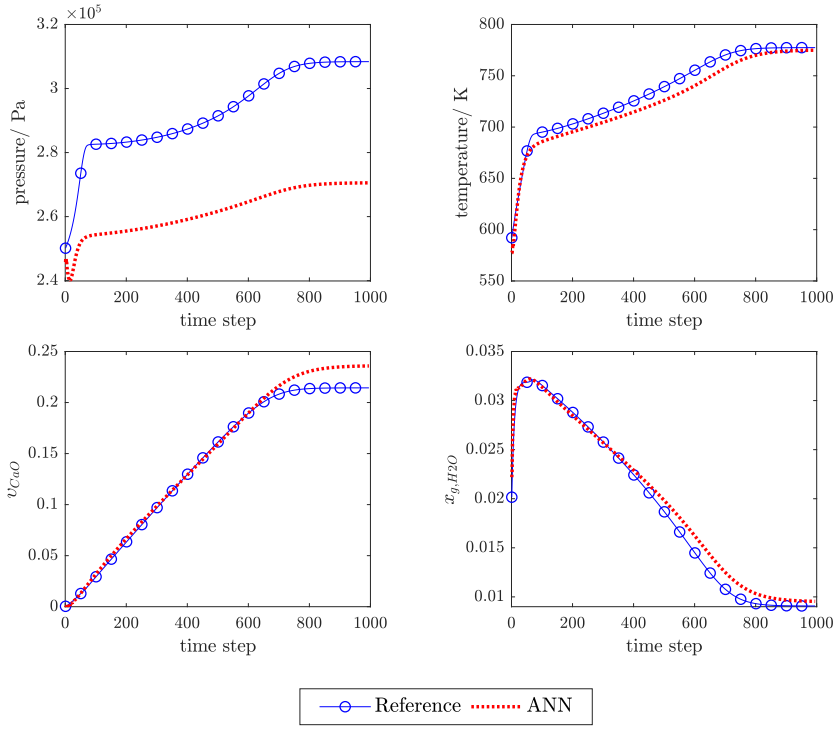


Figure 12. Worst prediction sample (red) obtained with "MSE + L2 + PHY" regularisation method and reference solution obtained from the physical model (blue).

4. Conclusions

We adopted a PINN framework as an example of grey-box modelling to predict the dynamic internal states of the TCES system. Our approach aligns with the motivation of PINN that sees the modelled system as a complex unit that is underrepresented by the governing equations used

in the physical model. We do not construct the ANN only as a surrogate model for the expensive numerical model, unlike in other PINN approaches that use ANN to infer the governing equations of the modelled system or the parameter values, assuming that the physical model describes the system perfectly. Our method was designed for application with real observation data that imply more complex processes than the simplified physical model. In this paper, however, we used synthetic data to train the ANN as a proof of concept.

As a key contribution, we propose to implement physical knowledge about a system into building the structure, choosing the training mode and designing the regularisation of ANNs to assure the physical plausibility and to increase the performance of the TCES dynamic internal state predictions. The alignment between the system's behaviour (dynamic and nonlinear) and the ANN structure is described. The ANN predictions using different regularisation strategies are also compared to show the improvement provided by our method.

We show that, while training in P structure is computationally more expensive and unstable, the result is superior to training using SP structure, because P training resembles the dynamic of the governing differential equations better. Additionally, we found that, due to the nonlinearity and long-term memory effects implied by the system equations, deeper recurrency is necessary. A moderate depth of feedback delay produces better prediction performance, resulting from the network ability to capture the latency of the system. However, using too much feedback delay is also counter-productive, as it does not give significant improvement anymore, only increasing computational cost.

We also show that including physical regularisation to train the network improves the physical plausibility of the network predictions, even for worst-case scenarios. Physical regularisation helps the network to learn about relationships between different input and target variables, as well as the time-dependency between them. This includes mole and energy balance equations that serve as the building blocks of the system's behaviour, along with simple monotonicity and non-negative constraints. However, physical regularisation alone is not enough to improve the generalisation capability of the network, and therefore, L2 regularisation is also necessary.

A very common issue with using ANNs in data-driven modelling is that obtaining experimental or operational data can be very costly, and therefore, there is often no sufficient data available to train the ANN. Our work shows that even with only a relatively small amount of training data (compared to the number of network parameters), using P training with a moderate amount of feedback delay d_y, combined with physical regularisation helps to prevent overfitting in optimising ill-posed problems and it produces relatively accurate and physically plausible predictions of the $CaO/Ca(OH)_2$ TCES system internal states. Further work is required for more sophisticated analysis of the system, for example with spatial distribution of the internal system, dynamic exogeneous input and uncertainty quantification of the predictions.

Availability of Data and Materials: Input-output data pairs used to train, validate, and test the ANN are available as supplementary materials on https://doi.org/10.18419/darus-633, while the trained network parameters (weight and bias values) for different regularisation methods explained in Section 3.3 are also available on https://doi.org/10.18419/darus-634. Each dataset is accompanied with a 'README' file that explains the data format.

Author Contributions: Conceptualisation, T.P., T.W., S.O. and W.N.; methodology, T.P., T.W., S.O. and W.N.; software, T.P. and T.W.; validation, T.P. and T.W.; formal analysis, T.P., T.W., S.O. and W.N.; investigation, T.P. and T.W.; resources, T.P. and T.W.; data curation, T.P.; writing–original draft preparation, T.P. and T.W.; writing–review and editing, S.O. and W.N.; visualisation, T.P. and T.W.; supervision, T.P., S.O. and W.N.; project administration, S.O. and W.N.; funding acquisition, S.O. and W.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC-2075–390740016.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Abbreviations:

ANN	Artificial Neural Network
BR	Bayesian Regularisation
LM	Levenberg Marquardt
MSE	Mean Squared Error
NARX	Nonlinear Autoregressive Network with Exogeneous Inputs
ODE	Ordinary Differential Equation
P	Parallel (network structure)
PDE	Partial Differential Equation
PINN	Physics Inspired Neural Network
RNN	Recurrent Neural Network
SP	Series Parallel (network structure)
TCES	Thermochemical Energy Storage

TCES-related parameters:

ΔH	Reaction enthalpy
λ_{eff}	Average thermal conductivity
μ	Viscosity
ν_s	Solid volume fraction
ϕ	Porosity
ρ_m	Mass density
ρ_n	Molar density
t	Time
c_p	Specific heat capacity
D	Effective diffusion coefficient
h	Specific enthalpy
K	Permeability
k_R	Reaction constant
p	Pressure
q	Source/sink term
T	Temperature
u	Specific internal energy
x_g	Gas molar fraction

ANN-related parameters:

α	Normalising constant of L2 regularisation term
β	Normalising constant of data-related error
$\boldsymbol{H}$	Hessian matrix
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{J}$	Jacobian matrix
$\hat{y}(t)$	Predicted value at time t
λ	Normalising constant of physical error
μ	Damping parameter for LM algorithm
θ	Network parameter
b	Network bias
d_y	Feedback delay
E_D	Data-related error
E_θ	Mean squared value of network parameters
E_{phy}	Physical error
L_θ	Loss function
N	Number of network parameters
n	Number of training samples

n_t Number of time steps
u Exogeneous input
w Network weight
$y(t)$ Observed value at time t

Appendix A. List of Exogeneous Input and Its Distribution

Table A1 lists all of the exogeneous input with their corresponding distributions. The exogeneous input distributions are centred around the values taken from [20].

Table A1. Input distributions for exogenous inputs, with μ and σ being the mean and standard deviation used to generate the data, respectively; while the superscript D and H refer to the dehydration and hydration process, respectively.

Exogenous inputs with normal distribution					
u	Unit	μ^D	σ^D	μ^H	σ^H
ρ_{CaO}	kg/m^3	1656	25	1656	25
$\rho_{Ca(OH)_2}$	kg/m^3	2200	25	2200	25
p_{init},p_{out}	Pa	1×10^1	2.3×10^3	2×10^5	2.3×10^3
T_{init}	K	573.15	20	773.15	20
T_{in}	K	773.15	20	573.15	20
$\dot{n}_{N_2,in}$	mol/s.m	4.632	0.25	2.04	0.15
$\dot{n}_{H_2O,in}$	mol/s.m	0.072	0.01	1.782	0.15
Exogenous inputs with lognormal distribution					
u	Unit	μ^D	σ^D	μ^H	σ^H
K	mD	log(5×10^3)	0.525	$log(5 \times 10^3)$	0.525
k_R	-	log(0.05)	0.5	$log(0.2)$	0.5
Exogenous inputs with shifted and scaled beta distribution					
u	Unit	a	b	scale	shift
$c_{p,CaO}$	J/kg.K	7.1	2.9	300	700
$c_{p,Ca(OH)_2}$	J/kg.K	7.6	2.4	350	1250
λ_{CaO}	W/m.K	6.5	3.5	0.6	-
$\lambda_{Ca(OH)_2}$	W/m.K	6.5	3.5	0.6	-
ϕ	-	8.5	1.5	0.825	-
ΔH	J/mol	4.8	5.2	3×10^4	9×10^4
$x_{H_2O,init}$	-	76	85	-	-

Appendix B. Mole and Energy Balance Error

Physical constraints in $e_{phy,i,t,1}$ and $e_{phy,i,t,2}$ use mole and energy balance equation, respectively. Both balances are calculated in a simplified way discretized for time steps of 5 s with spatially averaged values and a local thermal equilibrium of the gas and the solid. For clarity, all the specific sample indicators in the training dataset i are omitted in this section.

The mole balance is formulated for H_2O (assuming that the density can be calculated with ideal gas law) with the in- and outflowing moles $n_{H20,in}$ and $n_{H20,out}$, the storage term in the gaseous phase $\Delta n_{H20,sto}$ and the source/sink term $\Delta n_{H20,q}$. The in- and outflowing moles of H_2O both are known values from the simulation or input data. The storage term $\Delta n_{H20,sto}$ can be calculated from the change in H_2O molar fraction, $\hat{x}_{g,H_2O}(t) - \hat{x}_{g,H_2O}(t-1)$ multiplied with the H_2O molar density and the pore volume. The complete definition is written as:

$$\Delta n_{H_2O,sto}(t) = \phi V(\hat{x}_{g,H_2O}(t) - \hat{x}_{g,H_2O}(t-1))\rho_{n,H_2O}(t). \quad \text{(A1)}$$

The source/sink term $\Delta n_{H20,q}$ is calculated with the molar amount of CaO formed. Based on the stoichiometry ratio, with every 1 mole of CaO forme, 1 mole of H_2O is also formed. The molar amount of

CaO is determined by the change in CaO volume fraction, $\hat{v}_{CaO}(t) - \hat{v}_{CaO}(t-1)$, multiplied with the molar density and the volume. The calculation for $\Delta n_{H20,q}$ is written as:

$$\Delta n_{H_2O,q}(t) = V(\hat{v}_{CaO}(t) - \hat{v}_{CaO}(t-1))\rho_{n,CaO}. \tag{A2}$$

Finally, Equations (A1) and (A2) are substituted into Equation (17).

For the energy balance formulation, the energy of the inflowing and outflowing gas Q_{in} and Q_{out} are also known from simulation or from input data. The energy storage in the gaseous phase is neglected as its contribution is negligible. Only the solid contribution is used in the calculation of ΔQ_{sto}. The solid energy change is calculated as the change in both CaO and $Ca(OH)_2$ mass multiplied by the temperature and specific heat capacity. The definition is written as:

$$\Delta Q_{sto}(t) = Q_{sto}(t) - Q_{sto}(t-1), \tag{A3}$$

where $Q_{sto}(t)$ is defined as:

$$\begin{aligned} Q_{sto}(t) = V[&\hat{v}_{CaO}(t) c_{p,CaO} \cdot \rho_{m,CaO} + \\ &(1 - \phi - \hat{v}_{CaO}(t)) c_{p,Ca(OH)_2} \cdot \rho_{m,Ca(OH)_2}] \hat{T}(t). \end{aligned} \tag{A4}$$

The source/sink term for the energy balance equation, ΔQ_q, is calculated based on the change in molar amount of CaO multiplied by the specific reaction enthalpy and subtracted with the volume expansion work. The negative sign corresponds to the definition in Equation (7). The calculation is written as:

$$\Delta Q_q(t) = -V\Big(\hat{v}_{CaO}(t) - \hat{v}_{CaO}(t-1)\Big)\rho_{n,CaO}\Big(\Delta H - \frac{\phi}{1-\phi}\hat{T}(t)R\Big). \tag{A5}$$

Equations (A3) and (A5) are then substituted into Equation (18).

Appendix C. Example of the Ann Prediction

Figure A1 shows the best prediction of the ANN using 2 hidden layers with 15 and 8 nodes at each layer using only MSE and L2 regularisation to define the loss function.

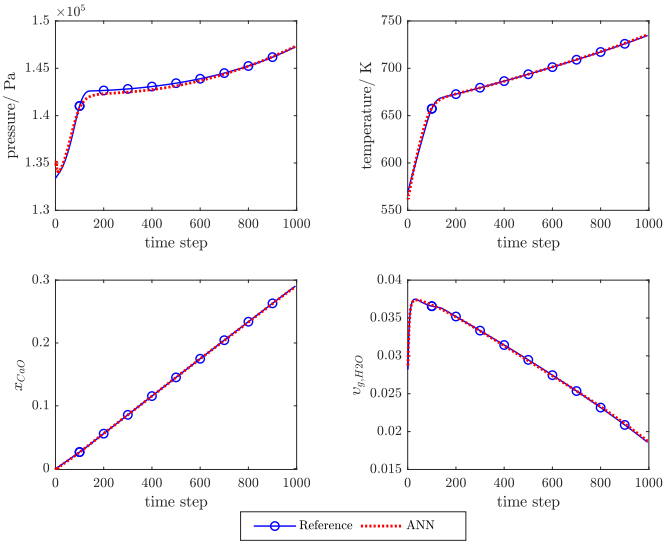


Figure A1. An example of the best prediction sample (red) obtained using 2 hidden layers with 15 and 8 nodes at each layer and reference solution obtained from the physical model (blue).

References

1. Haas, J.; Cebulla, F.; Cao, K.; Nowak, W.; Palma-Behnke, R.; Rahmann, C.; Mancarella, P. Challenges and trends of energy storage expansion planning for flexibility provision in power systems—A review. *Renew. Sustain. Energy Rev.* **2017**, *80*, 603–619. [CrossRef]
2. Møller, K.T.; Williamson, K.; Buckleyand, C.E.; Paskevicius, M. Thermochemical energy storage properties of a barium based reactive carbonate composite. *J. Mater. Chem.* **2020**, *8*, 10935–10942. [CrossRef]
3. Yuan, Y.; Li, Y.; Zhao, J. Development on Thermochemical Energy Storage Based on CaO-Based Materials: A Review. *Sustainability* **2018**, *10*, 2660. [CrossRef]
4. Pardo, P.; Deydier, A.; Anxionnaz-Minvielle, Z.; Rougé, S.; Cabassud, M.; Cognet, P. A review on high temperature thermochemical heat energy storage. *Renew. Sustain. Energy Rev.s* **2014**, *32*, 591–610. [CrossRef]
5. Scapino, L.; Zondag, H.; Van Bael, J.; Diriken, J.; Rindt, C. Energy density and storage capacity cost comparison of conceptual solid and liquid sorption seasonal heat storage systems for low-temperature space heating. *Renew. Sustain. Energy Rev.* **2017**, *76*, 1314–1331. [CrossRef]
6. Schaube, F.; Wörner, A.; Tamme, R. High Temperature Thermochemical Heat Storage for Concentrated Solar Power Using Gas-Solid Reactions. *J. Sol. Energy Eng.* **2011**, *133*, 7. [CrossRef]
7. Carrillo, A.; Serrano, D.; Pizarro, P.; Coronado, J. Thermochemical heat storage based on the Mn_2O_3/Mn_3O_4 redox couple: influence of the initial particle size on the morphological evolution and cyclability. *J. Mater. Chem.* **2014**, *2*, 19435–19443. [CrossRef]
8. Carrillo, A.; Moya, J.; Bayón, A.; Jana, P.; de la Peña O'Shea, V.; Romero, M.; Gonzalez-Aguilar, J.; Serrano, D.; Pizarro, P.; Coronado, J. Thermochemical energy storage at high temperature via redox cycles of Mn and Co oxides: Pure oxides versus mixed ones. *Sol. Energy Mater. Sol. Cells* **2014**, *123*, 47–57. [CrossRef]
9. Carrillo, A.; Sastre, D.; Serrano, D.; Pizarro, P.; Coronado, J. Revisiting the BaO_2/BaO redox cycle for solar thermochemical energy storage. *Phys. Chem. Chem. Phys.* **2016**, *18*, 8039–8048. [CrossRef]
10. J.P.Muthusamy.; Calvet, N.; Shamim, T. Numerical Investigation of a Metal-oxide Reduction Reactor for Thermochemical Energy Storage and Solar Fuel Production. *Energy Procedia* **2014**, *61*, 2054–2057. [CrossRef]
11. Block, T.; Knoblauch, N.; Schmücker, M. The cobalt-oxide/iron-oxide binary system for use as high temperature thermochemical energy storage material. *Thermochim. Acta* **2014**, *577*, 25–32. [CrossRef]
12. Michel, B.; Mazet, N.; Neveu, P. Experimental investigation of an innovative thermochemical process operating with a hydrate salt and moist air for thermal storage of solar energy: Global performance. *Appl. Energy* **2014**, *129*, 177–186. [CrossRef]
13. Uchiyama, N.; Takasu, H.; Kato, Y. Cyclic durability of calcium carbonate materials for oxide/water thermo-chemical energy storage. *Appl. Therm. Eng.* **2019**, *160*, 113893. [CrossRef]
14. Yan, T.; Wang, R.; Li, T.; Wang, L.; Fred, I. A review of promising candidate reactions for chemical heat storage. *Renew. Sustain. Energy Rev.* **2015**, *43*, 13–31. [CrossRef]
15. Zhang, H.; Baeyens, J.; Cáceres, G.; Degréve, J.; Lv, Y. Thermal energy storage: Recent developments and practical aspects. *Prog. Energy Combust. Sci.* **2016**, *53*, 1–40. [CrossRef]
16. André, L.; Abanades, S.; Flamant, G. Screening of thermochemical systems based on solid-gas reversible reactions for high temperature solar thermal energy storage. *Renew. Sustain. Energy Rev.* **2016**, *64*, 703–715. [CrossRef]
17. Schaube, F.; Koch, L.; Wörner, A.; Müller-Steinhagen, H. A thermodynamic and kinetic study of the de- and rehydration of $Ca(OH)_2$ at high H_2O partial pressures for thermo-chemical heat storage. *Thermochim. Acta* **2012**, *538*, 9–20. [CrossRef]
18. Schaube, F.; Kohzer, A.; Schütz, J.; Wörner, A.; Müller-Steinhagen, H. De- and rehydration of $Ca(OH)_2$ in a reactor with direct heat transfer for thermo-chemical heat storage. Part A: Experimental results. *Chem. Eng. Res. Des.* **2013**, *91*, 856–864. [CrossRef]
19. Schmidt, M.; Gutierrez, A.; Linder, M. Thermochemical energy storage with $CaO/Ca(OH)_2$ - Experimental investigation of the thermal capability at low vapor pressures in a lab scale reactor. *Appl. Energy* **2017**, *188*, 672–681. [CrossRef]
20. Shao, H.; Nagel, T.; Roßkopf, C.; Linder, M.; Wörner, A.; Kolditz, O. Non-equilibrium thermo-chemical heat storage in porous media: Part 2—A 1D computational model for a calcium hydroxide reaction system. *Energy* **2013**, *60*, 271–282. [CrossRef]

21. Nagel, T.; Shao, H.; Roßkopf, C.; Linder, M.; Wörner, A.; Kolditz, O. The influence of gas-solid reaction kinetics in models of thermochemical heat storage under monotonic and cyclic loading. *Appl. Energy* **2014**, *136*, 289–302. [CrossRef]
22. Bayon, A.; Bader, R.; Jafarian, M.; Fedunik-Hofman, L.; Sun, Y.; Hinkley, J.; Miller, S.; Lipiński, W. Techno-economic assessment of solid–gas thermochemical energy storage systems for solar thermal power applications. *Energy* **2018**, *149*, 473–484. [CrossRef]
23. Rezvanizaniani, S.M.; Liu, Z.; Chen, Y.; Lee, J. Review and recent advances in battery health monitoring and prognostics technologies for electric vehicle (EV) safety and mobility. *J. Power Sources* **2014**, *256*, 110–124. [CrossRef]
24. Mehne, J.; Nowak, W. Improving temperature predictions for Li-ion batteries: data assimilation with a stochastic extension of a physically-based, thermo-electrochemical model. *J. Energy Storage* **2017**, *12*, 288–296. [CrossRef]
25. Seitz, G.; Helmig, R.; Class, H. A numerical modeling study on the influence of porosity changes during thermochemical heat storage. *Appl. Energy* **2020**, *259*, 114152. [CrossRef]
26. Roßkopf, C.; Haas, M.; Faik, A.; Linder, M.; Wörner, A. Improving powder bed properties for thermochemical storage by adding nanoparticles. *Energy Convers. Manag.* **2014**, *86*, 93–98. [CrossRef]
27. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
28. Raissi, M.; Perdikaris, P.; Karniadakis, G. Inferring solutions of differential equations using noisy multi-fidelity data. *J. Comput. Phys.* **2017**, *335*, 736–746. [CrossRef]
29. Karamizadeh, S.; Abdullah, S.M.; Halimi, M.; Shayan, J.; j. Rajabi, M. Advantage and drawback of support vector machine functionality. In Proceedings of the 2014 International Conference on Computer, Communications, and Control Technology (I4CT), Langkawi, Malaysia, 2–4 September 2014; pp. 63–65.
30. Aggarwal, C. *Neural Networks and Deep Learning: A Textbook*, 1st ed.; Springer: Cham, Switzerland, 2018.
31. Oyebode, O.; Stretch, D. Neural network modeling of hydrological systems: A review of implementation techniques. *Nat. Resour. Model.* **2018**, *32*, e12189. [CrossRef]
32. Chen, S.; Wang, Y.; Tsou, I. Using artificial neural network approach for modelling rainfall–runoff due to typhoon. *J. Earth Syst. Sci.* **2013**, *122*, 399–405. [CrossRef]
33. Asadi, H.; Shahedi, K.; Jarihani, B.; Sidle, R.C. Rainfall-Runoff Modelling Using Hydrological Connectivity Index and Artificial Neural Network Approach. *Water* **2019**, *11*, 212. [CrossRef]
34. Wunsch, A.; Liesch, T.; Broda, S. Forecasting groundwater levels using nonlinear autoregressive networks with exogenous input (NARX). *J. Hydrol.* **2018**, *567*, 743–758. [CrossRef]
35. Taherdangkoo, R.; Tatomir, A.; Taherdangkoo, M.; Qiu, P.; Sauter, M. Nonlinear Autoregressive Neural Networks to Predict Hydraulic Fracturing Fluid Leakage into Shallow Groundwater. *Water* **2020**, *12*, 841. [CrossRef]
36. Kalogirou, S. Applications of artificial neural-networks for energy systems. *Appl. Energy* **1995**, *67*, 17–35. [CrossRef]
37. Bermejo, J.; Fernández, J.; Polo, F.; Márquez, A. A Review of the Use of Artificial Neural Network Models for Energy and Reliability Prediction. A Study of the Solar PV, Hydraulic and Wind Energy Sources. *Appl. Sci.* **2019**, *9*, 1844. [CrossRef]
38. Yaïci, W.; Entchev, E.; Longo, M.; Brenna, M.; Foiadelli, F. Artificial neural network modelling for performance prediction of solar energy system. In Proceedings of the 2015 International Conference on Renewable Energy Research and Applications (ICRERA), Palermo, Italy, 22–25 November 2015; pp. 1147–1151.
39. Kumar, A.; Zaman, M.; Goel, N.; Srivastava, V. Renewable Energy System Design by Artificial Neural Network Simulation Approach. In Proceedings of the 2014 IEEE Electrical Power and Energy Conference, Calgary, AB, Canada, 12–14 November 2014; pp. 142–147.
40. Breiman, L. Statistical Modeling: The Two Cultures. *Stat. Sci.* **2001**, *16*, 199–231. [CrossRef]
41. Zhang, Z.; Beck, M.W.; Winkler, D.A.; Huang, B.; Sibanda, W.; Goyal, H. Opening the black box of neural networks: methods for interpreting neural network models in clinical applications. *Ann. Transl. Med.* **2018**, *6*, 1–11. [CrossRef]
42. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10561.

43. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10566.
44. Doshi-Velez, F.; Kim, B. Towards A Rigorous Science of Interpretable Machine Learning *arXiv* **2017**, arXiv:1702.08608v2.
45. Miller, T. Explanation in Artificial Intelligence: Insights from the Social Sciences. *arXiv* **2017**, arXiv:1706.07269.
46. Karpatne, A.; Atluri, G.; Faghmous, J.; Steinbach, M.; Banerjee, A.; Ganguly, A.; Shekhar, S.; Samatova, N.; Kumar, V. Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data. *arXiv* **2017**, arXiv:1612.08544v2.
47. Tartakovsky, A.; Marrero, C.; Perdikaris, P.; Tartakovsky, G.; Barajas-Solano, D. Learning Parameters and Constitutive Relationships with Physics Informed Deep Neural Networks. *arXiv* **2018**, arXiv:1808.03398v2.
48. Karpatne, A.; Watkins, W.; Read, J.; Kumar, V. Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. *arXiv* **2018**, arXiv:1710.11431v2.
49. Wang, N.; Zhang, D.; Chang, H.; Li, H. Deep learning of subsurface flow via theory-guided neural network. *J. Hydrol.* **2020**, *584*, 124700. [CrossRef]
50. Chen, S.; Billings, S.; Grant, P. Non-linear system identification using neural networks. *Int. J. Control* **1990**, *51*, 1191–1214. [CrossRef]
51. Zhang, X. Time series analysis and prediction by neural networks. *Optim. Methods Softw.* **1994**, *4*, 151–170. [CrossRef]
52. Buitrago, J.; Asfour, S. Short-Term Forecasting of Electric Loads Using Nonlinear Autoregressive Artificial Neural Networks with Exogenous Vector Inputs. *Energies* **2017**, *10*, 40. [CrossRef]
53. Boussaada, Z.; Curea, O.; Remaci, A.; Camblong, H.; Bellaaj, N. A Nonlinear Autoregressive Exogenous (NARX) Neural Network Model for the Prediction of the Daily Direct Solar Radiation. *Energies* **2018**, *11*, 620. [CrossRef]
54. Mellit, A.; Kalogirou, S. Artificial intelligence techniques for photovoltaic applications: A review. *Prog. Energy Combust. Sci.* **2008**, *34*, 574–632. [CrossRef]
55. Jia, X.; Karpatne, A.; Willard, J.; Steinbach, M.; Read, J.; Hanson, P.; Dugan, H.; Kumar, V. Physics Guided Recurrent Neural Networks For Modeling Dynamical Systems: Application to Monitoring Water Temperature And Quality In Lakes *arXiv* **2018**, arXiv:1810.02880.
56. Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.* **1944**, *2*, 164–168. [CrossRef]
57. Marquardt, D. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 431–441. [CrossRef]
58. Yu, H.; B.M.Wilamowski. Levenberg-Marquardt Training. In *The Industrial Electronics Handbook—Intelligent Systems*, 2nd ed.; Wilamowski, B.; Irwin, J., Eds.; CRC Press: Boca Raton, FL, USA, 2011; Volume 5, Chapter 12.
59. Banerjee, I. Modeling Fractures in a $CaO/Ca(OH)_2$ Thermo-chemical Heat Storage Reactor. Master's Thesis, Universität Stuttgart, Stuttgart, Germany, 2018.
60. Koch, T.; Gläser, D.; Weishaupt, K.; Ackermann, S.; Beck, M.; Becker, B.; Burbulla, S.; Class, H.; Coltman, E.; Fetzer, T.; et al. *Release 3.0.0 of DuMu^x: DUNE for Multi-{Phase, Component, Scale, Physics, ...} Flow and Transport in Porous Media*; Zenodo: Geneva, Switzerland, 2019. [CrossRef]
61. Peinado, J.; Ibáñez, J.; Arias, E.; Hernández, V. Adams-Bashforth and Adams-Moulton methods for solving differential Riccati equations. *Comput. Math. Appl.* **2010**, *60*, 3032–3045. [CrossRef]
62. Tutueva, A.; Karimov, T.; Butusov, D. Semi-Implicit and Semi-Explicit Adams-Bashforth-Moulton Methods. *Mathematics* **2020**, *8*, 780. [CrossRef]
63. Beale, M.; Hagan, M.; Demuth, H. *Deep Learning Toolbox^{TM} User's Guide (R2019a)*; The MathWorks, Inc.: Natick, MA, USA, 2019.
64. Krogh, A.; Hertz, J.A. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems 4*; Moody, J.E.; Hanson, S.J.; Lippmann, R.P., Eds.; Denver, CO, USA, 2–5 December 1991 ; pp. 950–957.
65. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 15 April 2019).
66. MacKay, D. Bayesian Interpolation. *Neural Comput.* **1991**, *4*, 415–447. [CrossRef]
67. Sariev, E.; Germano, G. Bayesian regularized artificial neural networks for the estimation of the probability of default. *Quant. Financ.* **2020**, *20*, 311–328. [CrossRef]

68. Foresee, F.D.; Hagan, M. Gauss-Newton approximation to Bayesian learning. *IEEE* **1997**, *3*, 1930–1935. [CrossRef]
69. Nguyen, D.; Widrow, B. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *IEEE* **1990**, *3*, 21–26. [CrossRef]
70. Mittal, A.; Singh, A.P.; Chandra, P. A Modification to the Nguyen–Widrow Weight Initialization Method. In *Intelligent Systems, Technologies and Applications*; Springer: Singapore, 2020; pp. 141–153.
71. Zhang, G.; Patuwo, B.E.; Hu, M. Forecasting with artificial neural networks: The state of the art. *Int. J. Forecast.* **1998**, *14*, 35–62. [CrossRef]
72. Stathakis, D. Adams-Bashforth and Adams-Moulton methods for solving differential Riccati equations. *Int. J. Remote Sens.* **2009**, *30*, 2133–2147. [CrossRef]
73. Higham, D.J.; Trefethen, L.N. Stiffness of ODEs. *BIT Numer. Math.* **1993**, *33*, 285–303. [CrossRef]
74. Härter, F.P.; de Campos Velho, H.F.; Rempel, E.L.; Chian, A.C.L. Neural networks in auroral data assimilation. *J. Atmos. Sol.-Terr. Phys.* **2008**, *70*, 1243–1250. [CrossRef]
75. Awolusi, T.F.; Oke, O.L.; Akinkurolere, O.O.; Sojobi, A.O.; Aluko, O.G. Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete. *Heliyon* **2019**, *5*, 1–27. [CrossRef]

B Publication 2: Finite Volume Neural Network: Modeling Subsurface Contaminant Transport

FINITE VOLUME NEURAL NETWORK: MODELING SUBSURFACE CONTAMINANT TRANSPORT

Timothy Praditia *
University of Stuttgart

Matthias Karlbauer
University of Tübingen

Sebastian Otte
University of Tübingen

Sergey Oladyshkin
University of Stuttgart

Martin V. Butz
University of Tübingen

Wolfgang Nowak
University of Stuttgart

ABSTRACT

Data-driven modeling of spatiotemporal physical processes with general deep learning methods is a highly challenging task. It is further exacerbated by the limited availability of data, leading to poor generalizations in standard neural network models. To tackle this issue, we introduce a new approach called the Finite Volume Neural Network (FINN). The FINN method adopts the numerical structure of the well-known Finite Volume Method for handling partial differential equations, so that each quantity of interest follows its own adaptable conservation law, while it concurrently accommodates learnable parameters. As a result, FINN enables better handling of fluxes between control volumes and therefore proper treatment of different types of numerical boundary conditions. We demonstrate the effectiveness of our approach with a subsurface contaminant transport problem, which is governed by a non-linear diffusion-sorption process. FINN does not only generalize better to differing boundary conditions compared to other methods, it is also capable to explicitly extract and learn the constitutive relationships (expressed by the retardation factor). More importantly, FINN shows excellent generalization ability when applied to both synthetic datasets and real, sparse experimental data, thus underlining its relevance as a data-driven modeling tool.

1 INTRODUCTION

Training neural networks augmented with additional physical information has been shown to improve their generalization capabilities, particularly when predicting physical processes. In the Physics Informed Neural Network (PINN) framework (Karpatne et al., 2017; 2018; Tartakovsky et al., 2018; Raissi et al., 2019; Wang et al., 2020), the neural network prediction $u(x,t)$ is defined to be an explicit function of space x and time t. Furthermore, calculations of respective derivatives, such as $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial t}$, are required for formulating the loss function. However, when the available training data is concentrated on a single location x or time t, the approximation of the derivatives $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial t}$ in current techniques deteriorates due to (a) insufficient information provided by the data and (b) the lack of structural explainability of the framework itself. To address these issues from a structural point of view, several works have been conducted in the literature recently. One architecture, namely the Distributed Spatiotemporal Artificial Neural Network (DISTANA, Karlbauer et al., 2019), uses translational invariant Prediction Kernels (PKs) and Transition Kernels (TKs) to process the temporal and spatial correlation of the data, respectively. Another method, called the Universal Differential Equation method (UDE, Rackauckas et al., 2020), combines Convolutional Neural Networks (CNNs, LeCun et al., 1999) with Neural Ordinary Differential Equations (NODEs, Chen et al., 2018), for learning spatiotemporal data. Despite promising results shown by these methods, they still suffer from unreliable flux handling (i.e. the physical fluxes are not guaranteed to be conservative). Consequently, the methods mentioned above lack means to properly treat different types of boundary conditions.

To handle fluxes more robustly and improve generalization within a physical domain, we propose a Finite Volume Neural Network (FINN). The FINN method is a hybrid model, which capitalizes

*Corresponding author: timothy.praditia@iws.uni-stuttgart.de

on the structural knowledge of the well-known Finite Volume Method (FVM, Moukalled et al., 2016), and the flexibility as well as the learning abilities of Artificial Neural Networks (ANNs), more specifically NODEs. As a consequence, the FINN structure can properly treat different types of boundary conditions and ensures conservation of the quantities of interest. Moreover, we show that FINN is able to reconstruct the full field solution (for all x and t) even when trained with only partial information (e.g. at a single point x or t). Additionally, the structure of FINN facilitates learning and extracting constitutive relationships and/or reaction terms, and, consequently, shows exceptional generalization capabilities and develops explainable knowledge structures.

2 METHODS

In this work, we focus on modeling spatiotemporal physical processes, namely processes that scientists try to model with Partial Differential Equations (PDEs), such as diffusion-type problems. They can be generally written mathematically as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D(u)\frac{\partial u}{\partial x}\right) + q(u), \tag{1}$$

where u is the quantity of interest, t is time, D is the diffusion coefficient and q is the source/sink term. Usually, the FVM discretizes Equation 1 implicitly in space and explicitly in time, leading to a simplified definition:

$$\frac{\partial u_i^{(t+1)}}{\partial t} = f\left(u_{i-1}^{(t)}, u_i^{(t)}, u_{i+1}^{(t)}, t\right), \tag{2}$$

where $u_i^{(t)}$ denotes u at control volume i and time step t. In other words, the time derivative of u depends on the current value of u, the values of u at the neighboring control volumes, and time. For brevity, we drop the time index in later definitions.

In FINN, we introduce Flux Kernels $\mathcal{F}$, which take the input of u_{i-1}, u_i, and u_{i+1} to approximate the divergence part (first term on the right hand side) in Equation 1 for each control volume i:

$$\mathcal{F}_i = \Phi_\theta(u_{i-1}, u_i, u_{i+1}) = \sum_s fk_s \approx \oint_S \left(D(u)\frac{\partial u}{\partial x}\cdot\hat{n}\right) ds. \tag{3}$$

The Flux Kernels $\mathcal{F}_i$ consist of subkernels fk_s that calculate fluxes at the boundary surfaces s between control volume i and its neighboring control volumes (see Figure 1), being learned by $\mathcal{N}$ and $\mathcal{D}$, which are subcomponents of the function Φ with parameters θ. The function $\mathcal{N}$ in each fk_s is equivalent to a linear layer that takes u_i and one of its neighbors (i.e. u_{i-1} or u_{i+1}) as inputs and learns the numerical FVM stencil with its weights that should amount to $[-1, 1]$, which corresponds to $[u_i, u_{i-1}]$ and $[u_i, u_{i+1}]$ (i.e. simple difference between neighboring control volumes) in ideal one-dimensional diffusion problems. If the diffusion coefficient D depends on u according to a

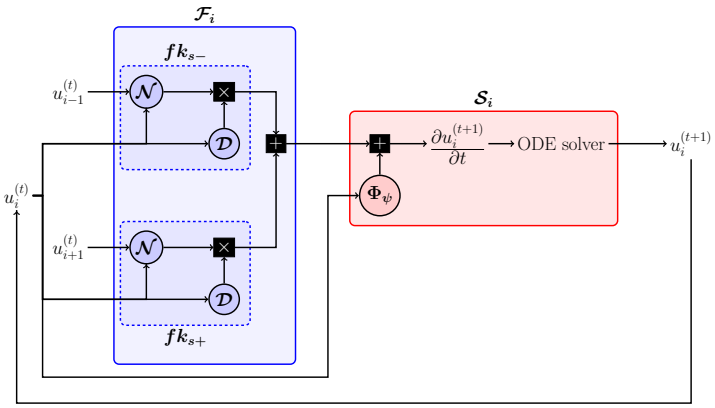


Figure 1: Illustration of the Flux and State Kernels in the FINN.

hidden constitutive relationship, the function $\mathcal{D}$ in each fk_s learns and approximates this function $\mathcal{D}(u) \approx D(u)$. Otherwise, $\mathcal{D}$ will be only a scalar value $\mathcal{D} \equiv D$, which can also be set as a learnable parameter. Next, the output of $\mathcal{N}$ is multiplied with the output of $\mathcal{D}$ to obtain the flux approximation at each boundary. When the fluxes at all boundary surfaces s are integrated in each control volume i, the summation of the numerical stencil will lead to the classical one-dimensional numerical Laplacian with $[1, -2, 1]$ corresponding to $[u_{i-1}, u_i, u_{i+1}]$ if Equation 1 is true.

This structure of the Flux Kernel enables straightforward handling of different types of boundary conditions. With a Dirichlet boundary condition $u = u_b$, we can set either $u_{i-1} = u_b$ or $u_{i+1} = u_b$ at the corresponding domain boundary. With a Neumann boundary condition ν, we can easily set the output of fk_s at the corresponding domain boundary s to be equal to ν. With a Cauchy boundary condition, we can calculate the derivative approximation at the corresponding domain boundary and set it as either u_{i-1} or u_{i+1}.

The State Kernels $\mathcal{S}$ then take the output of the Flux Kernels and approximate $\frac{\partial u}{\partial t}$ while also learning the source/sink term $q \approx \Phi_\psi(u)$ as a function of u (whenever necessary), which is also learnable by the State Kernels. Formally, each State Kernel can be written as an approximation of the time derivative in Equation 1 for each control volume i:

$$\mathcal{S}_i = \mathcal{F}_i + \Phi_\psi(u) \approx \frac{\partial u_i}{\partial t}, \tag{4}$$

where Φ is parameterized by ψ. The outputs of State Kernels are then integrated by an ODE solver to solve for $u^{(t+1)}$, which will be used recursively for calculation of the subsequent time steps. The benefits of using an ODE solver are twofold: (a) it allows for adaptive time stepping, which in turn leads to better numerical stability in explicit schemes, and (b) it enables handling of unevenly spaced time series, which is very common in real observation data.

One of the benefits of State Kernels is to enable separate calculations for different quantities of interest, while the divergence (flux) can be calculated based on the same variable. This ensures that each quantity of interest follows its own conservation law. In short, FINN consists of Flux Kernels that handle the spatial correlation, and State Kernels that handle the temporal correlation of the data.

3 Experiment

For demonstration purposes, we choose an application with a subsurface contaminant transport problem. We assess the performance of FINN not only using synthetic simulation data, but also real experimental data. The contaminant transport is characterized by the non-linear diffusion-sorption equation in a fluid-filled homogeneous porous medium:

$$R\frac{\partial c}{\partial t} = D_e\frac{\partial^2 c}{\partial x^2}, \tag{5}$$

where c denotes the concentration of trichloroethylene (TCE) dissolved in the fluid, D_e denotes the effective TCE diffusion coefficient, and R denotes the retardation factor (representing sorption), which is a function of c. Equation 5 is subject to a Dirichlet boundary condition at $x = 0$ and a Cauchy boundary condition at $x = L$ (i.e. the top and bottom ends of the field) and an initial condition $c(t = 0) = 0$. Using the definition of retardation factors, we can also calculate the total TCE concentration c_t (both in the fluid and adsorbed in the solid)

$$\frac{\partial c_t}{\partial t} = D_e\phi\frac{\partial^2 c}{\partial x^2}, \tag{6}$$

where ϕ is the porosity of the medium (i.e. the core samples). More detailed information on the experiment and its numerical simulation can be found elsewhere (Nowak & Guthke, 2016).

4 Results and Discussion

As the first step, we train and test FINN using numerically generated synthetic datasets. Both train and test datasets are discretized into 26 control volumes and 2 000 time steps. We train FINN using the whole spatial domain, with time steps $0-500$ (i.e. $t = 0-2\,500$ days) of the train dataset. Here,

Table 1: Comparison of MSE values between different deep learning architectures.

Method	Training	Extrapolated training	Test unseen	Parameters
TCN	$(7.9 \pm 5.4) \times 10^{-6}$	$(5.9 \pm 4.1) \times 10^{-3}$	$(3.0 \pm 1.2) \times 10^{-2}$	1 386
ConvLSTM	$(5.5 \pm 1.6) \times 10^{-6}$	$(4.9 \pm 5.7) \times 10^{-2}$	$(6.6 \pm 7.9) \times 10^{-2}$	1 496
DISTANA	$(1.9 \pm 1.1) \times 10^{-6}$	$(1.0 \pm 2.9) \times 10^{-2}$	$(1.6 \pm 4.0) \times 10^{-2}$	1 350
FINN	$(4.7 \pm 4.9) \times 10^{-5}$	$(1.1 \pm 1.2) \times 10^{-4}$	$(4.1 \pm 4.0) \times 10^{-5}$	528

FINN receives only the initial condition, i.e. initial values of $c^{(0)}$ and $c_t^{(0)}$, along with the Dirichlet boundary condition value at the top boundary. The bottom boundary is subject to a Cauchy boundary condition, and therefore is solution dependent. FINN is then trained in a closed loop system, using predicted values of c and c_t at time step t as input for the calculation at time step $t + 1$.

For this synthetic data application, we set $\mathcal{N}$ and $\mathcal{D}$ to be learnable. Additionally, for the calculation of c, we set $\mathcal{D}$ to be a feedforward neural network that approximates D_e/R in Equation 5, allowing us to extract information about the learned retardation factor as a function of the contaminant concentration $R(c)$. This neural network is constructed with 3 hidden layers, each containing 15 hidden nodes. Each hidden layer uses hyperbolic tangent as the activation function, and the output layer uses the sigmoid activation function, multiplied with a learnable scaling factor to ensure that the approximation of D_e/R is strictly positive.

To test the generalization capability, we use the trained network to extrapolate until time step 2 000 ($t = 10\,000$ days). Additionally, we test the trained FINN prediction against a completely unseen test dataset obtained at different boundary conditions. The Dirichlet boundary condition values at the top boundary c_s are 1.0 kg/m^3 and 0.7 kg/m^3 for the train and test dataset, respectively. We also compare the train and test Mean Squared Error (MSE) value with other known methods, such as TCN (Kalchbrenner et al., 2016), ConvLSTM (Shi et al., 2015), and DISTANA (Karlbauer et al., 2019).

The comparison in Table 1 shows that FINN appropriately generalizes when tested against a different boundary condition. Even though all methods have comparable performance during training, the

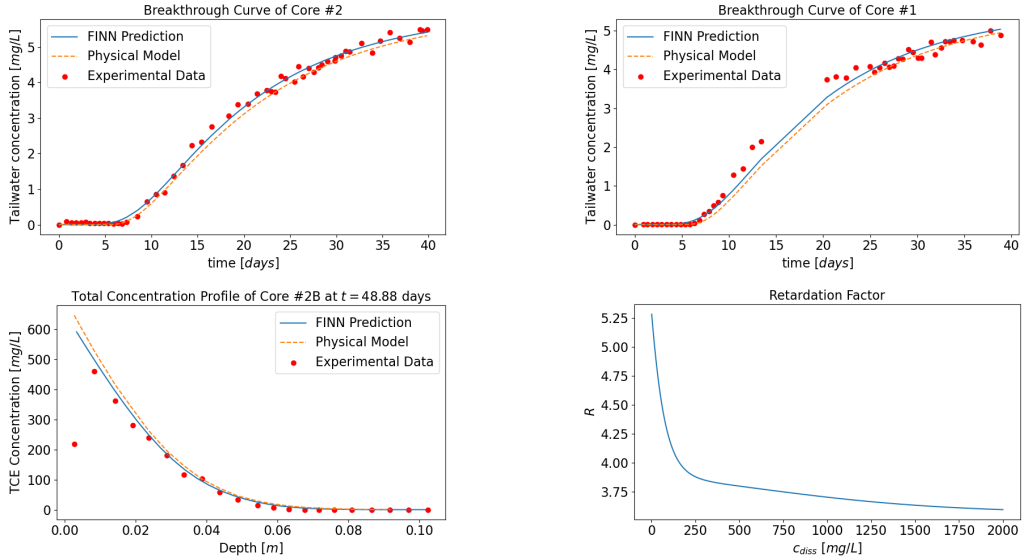


Figure 2: Breakthrough curve prediction of the FINN method (blue line) during training using data from core sample #2 (top left), during testing using data from core sample #1 (top right) and total concentration profile prediction using data from core sample #2B (bottom left). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The extracted retardation factor as a function of c is shown on the bottom right

predictions of TCN, ConvLSTM, and DISTANA deteriorate when the knowledge gained from the training data needs to be extrapolated and to predict unseen test data. This appears to be caused by the fact that they lack proper boundary condition handling. More detailed information can be found in the appendices regarding the benchmark test dataset (Appendix A), the training and test results (Appendix B), as well as the model setup (Appendix C).

As the second step, we apply FINN to real experimental data. The experimental data were collected from three different core samples, namely core samples #1, #2, and #2B (see Appendix D). In this setup, we train FINN using data that originate exclusively from core sample #2. For this experimental data application, FINN is set up the same way as the synthetic data setup. For the dissolved concentration calculation, we also set $\mathcal{D}$ to be a feedforward neural network that takes c as the input to learn the retardation factor function. However, we assume that we know the diffusion coefficient values for all core samples. The main setup difference lies in the available data used to train FINN. More specifically, we only use the breakthrough curve data of c in the tailwater. This means that the data provides partial information and constrains the FINN prediction only at $x = L|_{0 \leq t \leq t_{end}}$.

The results show that the trained FINN has higher accuracy with $\mathrm{MSE} = 4.84 \times 10^{-4}$ compared to the calibrated (least squares) numerical PDE model with $\mathrm{MSE} = 1.06 \times 10^{-3}$. Further, we test and validate the trained model using different core samples (i.e. #1 and #2B), which originate from the same geographical area and therefore can be assumed to have similar soil parameters. In Figure 2, we show that the predictions match the experimental data with reasonable accuracy. We also compare the predictions with the output of the numerical model, with the retardation factor also calibrated using the data from the same core sample #2. The plots show that FINN's prediction accuracy is comparable to the numerical model, even beating it in some instances. For core sample #1, the MSE of FINN prediction is 1.37×10^{-3}, while the numerical model underestimates the breakthrough curve with $\mathrm{MSE} = 2.50 \times 10^{-3}$.

Specifically for core sample #2B, which is significantly longer than the other samples, to model the diffusion-sorption process, we can assume a zero-flux Neumann boundary condition at the bottom of the core. As a consequence, there are no breakthrough curve data available anymore. Instead, we compare the prediction against the total concentration profile data obtained from a destructive sampling (i.e. core slicing) at the end of the experiment. Here, FINN produces predictions with $\mathrm{MSE} = 1.16 \times 10^{-3}$, while the numerical model overestimates the total concentration profile with $\mathrm{MSE} = 2.73 \times 10^{-3}$. The results show that, even when applied to a different type of boundary condition, FINN's predictions remain accurate. Moreover, we can extract the retardation factor as a function of c using FINN, which is plotted in Figure 2, thus explaining a soil property.

5 Conclusion and Future Work

We have shown that including physical knowledge in the form of the Finite Volume structure produces excellent generalization capabilities and improves the explainability of the applied neural network structure. Our novel Finite Volume Neural Network (FINN) permits proper calculations for conservative fluxes and for different types of boundary condition. FINN outperforms other neural network methods for spatiotemporal modeling such as Temporal Convolutional Network, Convolutional LSTM, and DISTANA, especially when tested with different boundary conditions. Moreover, we show that FINN is suitable for experimental data processing, rendering it relevant as a data-driven modeling tool.

In this work, we assumed spatial homogeneity for the soil in the simulation domain due to the small size of the actual experimental domain. For real field applications, where the scale is significantly larger, the homogeneity assumption might not hold. We are interested in enhancing FINN to handle spatially heterogeneous parameters. One way to achieve this is by defining a spatially correlated parameter to model a learnable diffusion coefficient in a spatially heterogeneous system akin to Karlbauer et al. (2020). Additionally, we are also interested in quantifying uncertainties of our model by implementing a Bayesian Neural Network. A concrete application example in the contaminant transport modeling domain is to produce confidence intervals for both the concentration function and the learned retardation factor function. Overall, recent development in fusing numerical methods with deep learning to aid physical processes simulation shows promising results to keep continuing the trend in this direction. It is very exciting to see how far we can push the boundaries between numerical methods and deep learning and to see the benefit when combining both approaches.

ACKNOWLEDGMENTS

This work is funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016 as well as EXC 2064 – 390727645. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). Moreover, we thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer. Codes and data that are used for this paper can be found in the repository https://github.com/timothypraditia/finn.

REFERENCES

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint*, 2016. URL https://arxiv.org/abs/1610.10099.

Matthias Karlbauer, Sebastian Otte, Hendrik P. A. Lensch, Thomas Scholten, Volker Wulfmeyer, and Martin V. Butz. A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1912.11141.

Matthias Karlbauer, Tobias Menge, Sebastian Otte, Hendrik P. A. Lensch, Thomas Scholten, Volker Wulfmeyer, and Martin V. Butz. Hidden latent state inference in a spatio-temporal generative model, 2020.

A. Karpatne, G. Atluri, J.H. Faghmous, M. Steinbach, A. Banerjee, A.R. Ganguly, S. Shekhar, N.F. Samatova, and V. Kumar. Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data. *arXiv preprint*, 2017. URL http://arxiv.org/abs/1612.08544v2.

A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. *arXiv preprint*, 2018. URL http://arxiv.org/abs/1710.11431v2.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. *Object recognition with gradient-based learning*, pp. 319–345. Springer, 1999. ISBN 978-3-540-46805-9. doi: 10.1007/3-540-46805-6_19.

F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. ISBN 9783319168746. doi: 10.1007/978-3-319-16874-6.

W. Nowak and A. Guthke. Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, 18(11), 2016. doi: 10.3390/e18110409.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2001.04385.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint*, 2015. URL https://arxiv.org/abs/1506.04214.

A.M. Tartakovsky, C.O. Marrero, P. Perdikaris, G.D. Tartakovsky, and D. Barajas-Solano. Learning Parameters and Constitutive Relationships with Physics Informed Deep Neural Networks. *arXiv preprint*, 2018. URL https://arxiv.org/abs/1808.03398.

Nanzhe Wang, Dongxiao Zhang, Haibin Chang, and Heng Li. Deep learning of subsurface flow via theory-guided neural network. *Journal of Hydrology*, 584:124700, 2020. doi: 10.1016/j.jhydrol.2020.124700.

A SOIL PARAMETERS AND SIMULATION DOMAINS FOR THE BENCHMARK TEST

In this appendix, we present the soil parameters and the simulation domains used to generate the numerical benchmark dataset. Table 2 summarizes the parameters used to generate the training and test dataset. The retardation factor function is generated with the Freundlich sorption isotherm, written mathematically as

$$R = 1 + \frac{1-\phi}{\phi}\rho_s K_f n_f c^{n_f - 1}. \tag{7}$$

Table 2: Soil parameters and simulation domains for training and testing dataset generation.

Soil parameters			Simulation domain		
Parameter	Unit	Value	Parameter	Unit	Value
D_e	m^2/day	5×10^{-4}	L	m	1.0
ϕ	-	0.29	Δx	m	0.04
ρ_s	kg/m^3	2880	t_{end}	days	10^4
K_f	(m^3/kg)^{n_f}	3.53×10^{-4}	Δt	days	5
n_f	-	0.874			

Here, D_e is the effective diffusion coefficient, ϕ is the porosity, ρ_s is the bulk density, K_f is the Freundlich K coefficient, n_f is the Freundlich exponent, L is the length of the sample, Δx is the discrete control volume size, t_{end} is the simulation time, and Δt is the numerical time step.

For the training dataset, the upper boundary condition ($x = 0$) is set to be a Dirichlet boundary condition, with the maximum solubility of TCE $c_s = 1.0$ kg/m^3. The testing dataset is generated with the same soil parameters and simulation domain, but with upper boundary condition $c_s = 0.7$ kg/m^3. The lower boundary condition ($x = L$) is set to be a Cauchy boundary condition according to $\frac{D_e}{Q}\frac{\partial c}{\partial x}|_{x=L}$, where Q is the flow rate in the bottom reservoir. In the benchmark dataset, we assume that $Q = 1.0$. Details on geometries, boundary conditions, and simulation can be found in (Nowak & Guthke, 2016).

B BENCHMARK TEST RESULTS

In this appendix, we present the results and compare different methods for the benchmark test results. For each method, we train 10 models with different initialization. The MSE values of the predictions are then calculated compared with the training dataset at time steps $0-500$ (i.e. $t = 0-2\,500$ days), the extrapolated training dataset at time steps $500 - 2\,000$ (i.e. $t = 2\,500 - 10\,000$ days), and the whole unseen test dataset (at all time steps $0 - 2\,000$). We train the models with noisy data. The noise is normally distributed with standard deviation $\sigma = 1 \times 10^{-5}$, i.e. $\mathcal{N} \sim (0.0, 1 \times 10^{-5})$. Detailed information of the test MSE for every individual model is shown in Table 3 for seen data and in Table 4 for unseed data.

The prediction mean and confidence interval are plotted in Figure 3, Figure 4, Figure 5, and Figure 6. Confidence intervals are obtained from repeated (ten times) training with random initialization. Even though the prediction mean of each method is not far from the synthetic data, clear instabilities and inconsistencies can be seen from the wide range of confidence intervals in the TCN, ConvLSTM, and DISTANA predictions. This instability is mainly caused by the improper handling of boundary conditions by these methods. FINN, on the other hand, produces very precise prediction along with high accuracy.

C MODEL DETAILS OF TCN, CONVLSTM AND DISTANA

In this appendix, we provide additional information about the TCN, ConvLSTM and DISTANA models: bias neurons were used in all layers of all architectures and ADAM was used for optimization with a learning rate of $\eta = 1 \times 10^{-3}$. As fair comparison, FINN results for the benchmark test

Table 3: Test MSE on seen data (extrapolated training) from ten different training runs for each model

Run	TCN	ConvLSTM	DISTANA	FINN
1	5.2×10^{-3}	2.1×10^{-3}	6.3×10^{-5}	2.7×10^{-4}
2	4.9×10^{-3}	3.4×10^{-3}	3.6×10^{-4}	3.0×10^{-5}
3	4.1×10^{-3}	8.3×10^{-2}	9.7×10^{-2}	2.7×10^{-4}
4	4.1×10^{-3}	1.5×10^{-1}	4.0×10^{-4}	9.0×10^{-5}
5	8.1×10^{-4}	4.4×10^{-3}	4.2×10^{-5}	8.6×10^{-6}
6	1.2×10^{-2}	8.3×10^{-3}	4.5×10^{-4}	4.1×10^{-5}
7	1.5×10^{-2}	2.4×10^{-3}	8.2×10^{-5}	3.2×10^{-5}
8	4.5×10^{-3}	5.0×10^{-3}	9.5×10^{-4}	2.8×10^{-4}
9	2.6×10^{-3}	1.0×10^{-1}	5.2×10^{-5}	2.4×10^{-5}
10	5.6×10^{-3}	1.3×10^{-1}	1.8×10^{-4}	3.5×10^{-5}

Table 4: Test MSE on unseen data coming from ten different training runs for each model

Run	TCN	ConvLSTM	DISTANA	FINN
1	3.8×10^{-2}	1.1×10^{-2}	1.5×10^{-3}	9.7×10^{-5}
2	3.3×10^{-2}	1.1×10^{-3}	8.9×10^{-4}	1.5×10^{-5}
3	3.0×10^{-2}	1.0×10^{-1}	1.4×10^{-1}	9.5×10^{-5}
4	2.7×10^{-2}	1.2×10^{-1}	8.6×10^{-3}	3.4×10^{-5}
5	2.5×10^{-2}	7.0×10^{-3}	7.0×10^{-5}	4.9×10^{-6}
6	5.1×10^{-2}	5.6×10^{-4}	3.6×10^{-3}	1.9×10^{-5}
7	2.9×10^{-2}	2.6×10^{-2}	3.0×10^{-4}	1.5×10^{-5}
8	3.9×10^{-3}	3.1×10^{-4}	8.6×10^{-3}	1.0×10^{-4}
9	2.3×10^{-2}	1.9×10^{-1}	3.4×10^{-3}	1.2×10^{-5}
10	4.3×10^{-2}	2.2×10^{-1}	3.7×10^{-4}	1.6×10^{-5}

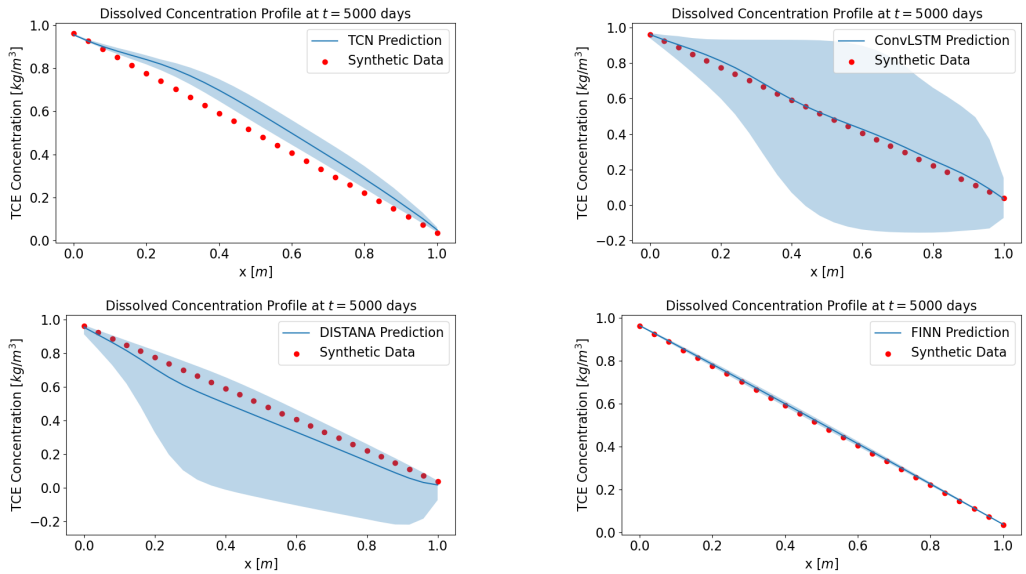


Figure 3: Dissolved concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the extrapolated training dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

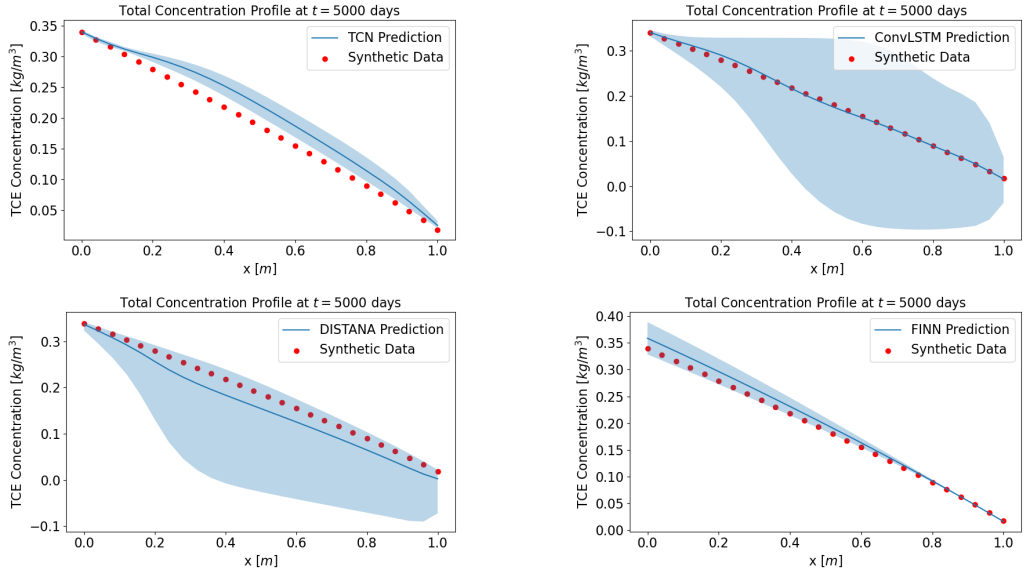


Figure 4: Total concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the extrapolated training dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

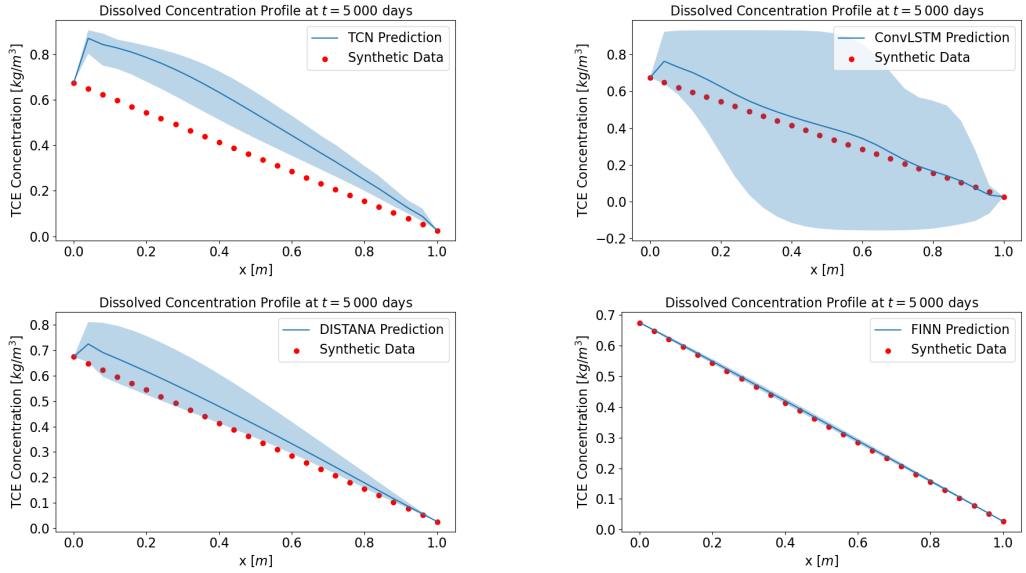


Figure 5: Dissolved concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the test dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

case are obtained also using the ADAM optimizer with the same learning rate. While TCN, ConvLSTM and DISTANA are always provided with the real data in the first ten timesteps (i.e. teacher forcing), FINN only receives an initial condition in the first timestep along with the information about the boundary in all timesteps. For better comparison, we also provide boundary condition information for TCN, ConvLSTM, and DISTANA. Note that in this experiment, TCN, ConvLSTM and DISTANA are provided with more information than FINN, which nevertheless outperforms the other models.

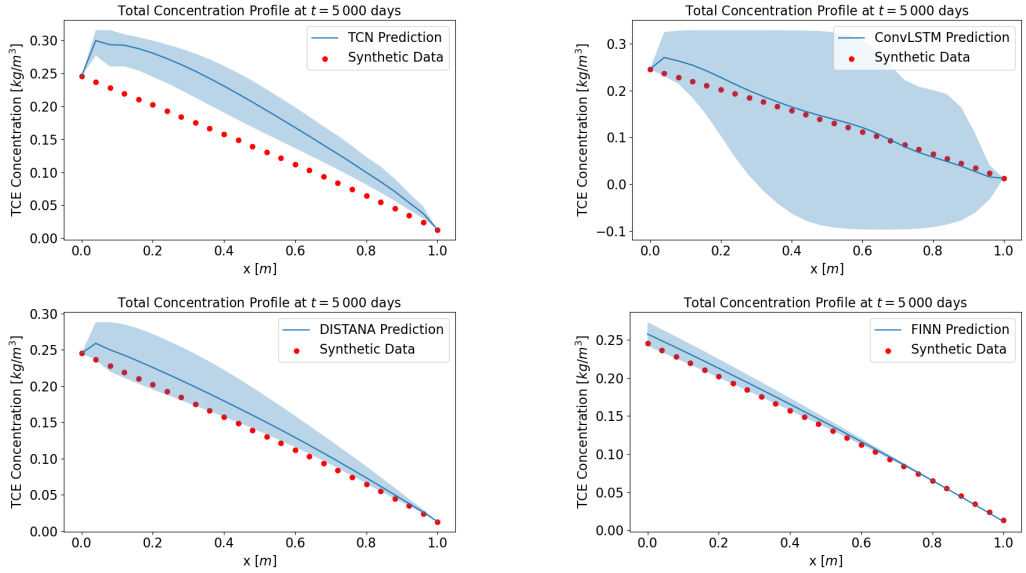


Figure 6: Total concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the test dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

TCN Two input channels are followed by two hidden layers with four and eight channels, respectively, which are processed by two output channels. A convolution kernel size of $k = 3$ was chosen and the standard dilation rate of TCN was applied ($d = l^2$, where l is the index of the layer), leading to a time horizon of 28 time steps. Code was taken and modified from [1].

ConvLSTM Two input feature maps, followed by ten channels in one hidden layer and two output channels, were used. The convolution kernel size was set to $k = 3$ and zero-padding was applied to preserve data dimensions. PyTorch code was taken from [2] and adapted to be applicable to spatially one-dimensional data.

DISTANA Two input channels map to four preprocessing convolution channels, which feed forward into a ConvLSTM layer with eight channels which are processed by two postprocessing convolution channels. The lateral information processing convolution layer was set to two channels.

D Soil Parameters and Simulation Domains for the Experiment

In this appendix, we present the soil parameters and the simulation domains of the core samples used in the experiment. Table 5 summarizes the parameters of core sample #1, #2, and #2B.

For all experiments, the core samples are subjected to a constant TCE concentration at the top c_s, which amounts to a Dirichlet boundary condition. Notice that, for core sample #2, we set c_s to be slightly higher to compensate for the fact that there might be fractures at the top of core sample #2, so that the TCE can break through the core sample faster.

For core samples #1 and #2, Q is the flow rate of clean water at the bottom reservoir that determines the Cauchy boundary condition at the bottom of the core samples. For core sample #2B, note that the sample length is significantly longer than the other samples. Therefore, for this particular sample, given t_{end} to be approximately in the same order with the other samples, we assume the bottom boundary condition to be a no-flow Neumann boundary condition.

[1] https://github.com/locuslab/TCN

[2] https://github.com/ndrplz/ConvLSTM_pytorch

Table 5: Soil and experimental parameters of core samples #1, #2, and #2B.

Soil parameters				
Parameter	Unit	Core #1	Core #2	Core #2B
D_e	m^2/day	2.00×10^{-5}	2.00×10^{-5}	2.78×10^{-5}
ϕ	-	0.288	0.288	0.288
ρ_s	kg/m^3	1957	1957	1957
Simulation domain				
Parameter	Unit	Core #1	Core #2	Core #2B
L	m	0.0254	0.02604	0.105
r	m	0.02375	0.02375	N/A
t_{end}	days	38.81	39.82	48.88
Q	m^3/day	1.01×10^{-4}	1.04×10^{-4}	N/A
c_s	kg/m^3	1.4	1.6	1.4

C Publication 3: Composing Partial Differential Equations with Physics-Aware Neural Networks

Composing Partial Differential Equations with Physics-Aware Neural Networks

Matthias Karlbauer[⋆ 1] **Timothy Praditia**[⋆ 2] **Sebastian Otte**[1] **Sergey Oladyshkin**[2] **Wolfgang Nowak**[2] **Martin V. Butz**[1]

Abstract

We introduce a compositional physics-aware neural network (FINN) for learning spatiotemporal advection-diffusion processes. FINN implements a new way of combining the learning abilities of artificial neural networks with physical and structural knowledge from numerical simulation by modeling the constituents of partial differential equations (PDEs) in a compositional manner. Results on both one- and two-dimensional PDEs (Burgers', diffusion-sorption, diffusion-reaction, Allen-Cahn) demonstrate FINN's superior modeling accuracy and excellent out-of-distribution generalization ability beyond initial and boundary conditions. With only one tenth of the number of parameters on average, FINN outperforms pure machine learning and other state-of-the-art physics-aware models in all cases—often even by multiple orders of magnitude. Moreover, FINN outperforms a calibrated physical model when approximating sparse real-world data in a diffusion-sorption scenario, confirming its generalization abilities and showing explanatory potential by revealing the unknown retardation factor of the observed process.

1. Introduction

Artificial neural networks (ANNs) are considered universal function approximators (Cybenko, 1989). Their effective learning ability, however, greatly depends on domain and task-specific prestructuring and methodological modifications referred to as inductive biases (Battaglia et al., 2018). Typically, inductive biases limit the space of possible models by reducing the opportunities for computational shortcuts, which can lead to erroneous implications derived from a potentially limited dataset (overfitting). The recently evolving field of physics-informed machine learning employs physical knowledge as inductive bias providing vast generalization advantages in contrast to pure machine learning (ML) in physical domains (Raissi et al., 2019). While numerous approaches have been introduced to augment ANNs with physical knowledge, these methods either do not allow the incorporation of explicitly defined physical equations (Long et al., 2018; Seo et al., 2019; Guen & Thome, 2020; Li et al., 2020a; Sitzmann et al., 2020) or cannot generalize to other initial and boundary conditions than those encountered during training (Raissi et al., 2019).

In this work, we present the finite volume neural network (FINN) model—a physics-aware neural network structure adhering to the idea of spatial and temporal discretization in numerical simulation. FINN consists of multiple neural network modules that interact in a distributed, compositional manner (Battaglia et al., 2018; Lake et al., 2017; Lake, 2019). The modules are designed to account for specific parts of advection-diffusion equations, a class of partial differential equations (PDEs). This modularization allows to combine two advantages that are not yet met by state-of-the-art models: the explicit incorporation of physical knowledge and the generalization over initial and boundary conditions. To the best of our knowledge, FINN's ability to adjust to different initial and boundary conditions and to explicitly learn constitutive relationships and reaction terms is unique, yielding excellent out-of-distribution generalization. The core contributions of this work are:

- Introduction of FINN, a physics-aware neural network model, explicitly designed to generalize over initial and boundary conditions, demonstrating excellent generalization ability.
- Evaluation of state-of-the-art pure ML and physics-aware models, contrasted to FINN on one- and two-dimensional benchmarks, demonstrating benefits of explicit model design.
- Application of FINN to a real-world contamination-diffusion problem, verifying its applicability to real, spatially and temporally constrained training data.

[⋆]Equal contribution [1]Neuro-Cognitive Modeling, University of Tübingen, Tübingen, Germany [2]Department of Stochastic Simulation and Safety Research for Hydrosystems, University of Stuttgart, Stuttgart, Germany. Correspondence to: Matthias Karlbauer <matthias.karlbauer@uni-tuebingen.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

2. Related Work

Non-Physics-Aware ANN Architectures Pure ML models that are designed for spatiotemporal data processing can be separated into temporal convolution (TCN, Kalchbrenner et al., 2016) and recurrent neural networks. While the former perform convolutions over space and time, representatives of the latter, e.g., convolutional LSTM (ConvLSTM, Shi et al., 2015) or DISTANA (Karlbauer et al., 2019), aggregate spatial neighbor information to further process the temporal component with recurrent units. Since pure ML models do not adhere to physical principles, they require large amounts of training data and parameters in order to approximate a desired physical process; but still are not guaranteed to behave consistently outside the regime of the training data.

Physics-Aware ANN Architectures When designed to satisfy physical equations, ANNs are reported to have greater robustness in terms of physical plausibility. For example, the physics-informed neural network (PINN, Raissi et al., 2019) consists of an MLP that satisfies an explicitly defined PDE with specific initial and boundary conditions, using automatic differentiation. However, the remarkable results beyond the time steps encountered during training are limited to the very particular PDE and its conditions. A trained PINN cannot be applied to different initial conditions, which limits its applicability in real-world scenarios.

Other methods by Long et al. (PDENet, 2018), Guen & Thome (PhyDNet, 2020), or Sitzmann et al. (SIREN, 2020) learn the first n derivatives to achieve a physically plausible behavior, but lack the option to include physical equations. The same limitation holds when operators are learned to approximate PDEs (Li et al., 2020a;b), or when physics-aware graph neural networks are applied (Seo et al., 2019). Yin et al. (APHYNITY, 2020) suggest to approximate equations with an appropriate physical model and to augment the result by an ANN, preventing the ANN to approximate a distinct part within the physical model. For more comparison to related work, please refer to subsection B.1.

In summary, none of the above methods can explicitly learn particular constitutive relationships or reaction terms while simultaneously generalizing beyond different initial and boundary conditions.

3. Finite Volume Neural Network (FINN)

Problem Formulation Here, we focus on modeling spatiotemporal physical processes. Specifically, we consider systems governed by advection-diffusion type equations (Smolarkiewicz, 1983):

$$\frac{\partial u}{\partial t} = D(u)\frac{\partial^2 u}{\partial x^2} - v(u)\frac{\partial u}{\partial x} + q(u), \tag{1}$$

where u is the quantity of interest, t is time, x is the spatial coordinate, D is the diffusion coefficient, v is the advection velocity, and q is the source/sink term. Eq. 1 can be partitioned into three parts: the storage term, the flux terms, and the source/sink term. The storage term $\frac{\partial u}{\partial t}$ describes the change of the quantity u over time. The flux terms are the advective flux $v(u)\frac{\partial u}{\partial x}$ and the diffusive flux $D(u)\frac{\partial^2 u}{\partial x^2}$. Both calculate the amount of u exchanged between neighboring volumes. The source/sink term $q(u)$ describes the generation or elimination of the quantity u. Eq. 1 is a general form of PDEs with up to second order spatial derivatives, but it has a wide range of applicability due to the flexibility of defining $D(u)$, $v(u)$, and $q(u)$ as different functions of u, as is shown by the numerical experiments in this work.

The finite volume method (FVM, Moukalled et al., 2016) discretizes a simulation domain into control volumes ($i = 1, \dots, N_x$), where exchange fluxes are calculated using a surface integral (Riley et al., 2009). In order to match this structure in FINN, we introduce two different kernels, which are (spatially) casted across the discretized control volumes: the flux kernel, modeling the flux terms (i.e. lateral quantity exchange), and the state kernel, modeling the source/sink term as well as the storage term. The overall FINN architecture is shown in Figure 1.

Flux Kernel The flux kernel $\mathcal{F}$ approximates the surface integral for each control volume i with boundary Ω by a composition of multiple subkernels f_j, each representing the flux through a discretized surface element j:

$$\mathcal{F}_i = \sum_{j=1}^{N_{s_i}} f_j \approx \oint_{\omega \subseteq \Omega} \left(D(u)\frac{\partial^2 u}{\partial x^2} - v(u)\frac{\partial u}{\partial x} \right) \cdot \hat{n}\, d\Gamma, \tag{2}$$

where N_{s_i} is the number of discrete surface elements of control volume i, ω is a continuous surface element (a subset of Ω), f_j are subkernels (consisting of feedforward network modules), and $\hat{n}$ is the unit normal vector pointing outwards of ω.

In our exemplary one-dimensional arrangement, two subkernels f_{i-} and f_{i+} (see Figure 1) contain the modules $\varphi_{\mathcal{D}}$, $\varphi_{\mathcal{A}}$, and $\varphi_{\mathcal{N}}$. Module $\varphi_{\mathcal{N}}$ is a linear layer with the purpose to approximate the first spatial derivative $\frac{\partial u}{\partial x}$, i.e.

$$\frac{\partial u_i}{\partial x} \approx \begin{cases} \varphi_{\mathcal{N}}(u_i, u_{i-1}) & \text{on } f_{i-} \\ \varphi_{\mathcal{N}}(u_i, u_{i+1}) & \text{on } f_{i+} \end{cases}. \tag{3}$$

Technically, $\varphi_{\mathcal{N}}$ is supposed to learn the numerical FVM stencil, being nothing else but the difference between its inputs, i.e. the quantity at two neighboring control volumes in ideal one-dimensional problems. This signifies that the weights of $\varphi_{\mathcal{N}}$ should amount to $[-1, 1]$ with respect to $[u_i, u_{i-1}]$ and $[u_i, u_{i+1}]$ in order to output their difference.

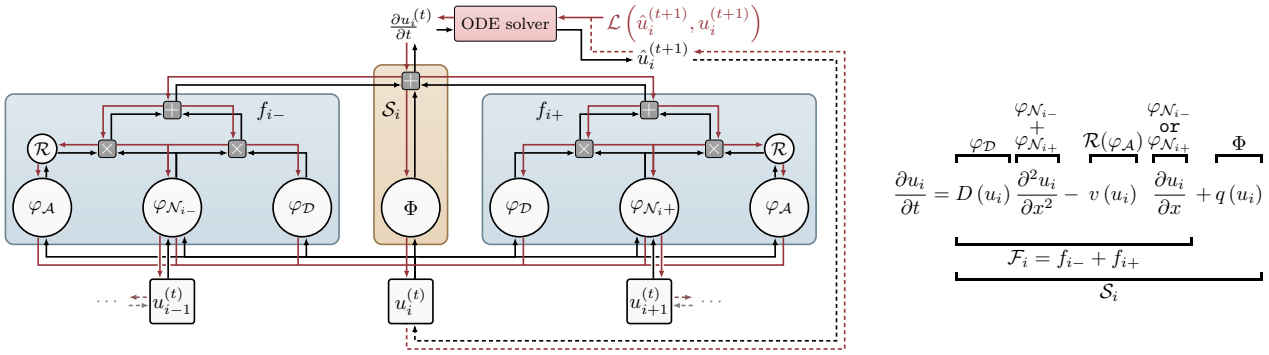


Figure 1: Flux and state kernels in FINN for the one-dimensional case (left); red lines indicate gradient flow. And detailed assignment of the individual modules with their contribution to Eq. 1 (right).

Module $\varphi_{\mathcal{D}}$, receiving u_i as input, is responsible for diffusive flux (the process of quantity homogenization from areas of high to low concentration). In case the diffusion coefficient D depends on u, the module is designed as feedforward network, such that $\varphi_{\mathcal{D}}(u) \approx D(u)$. Otherwise, $\varphi_{\mathcal{D}}$ is set as scalar value, which can be set trainable if the value of D is unknown.

Things are more complicated for advective flux, representing bulk motion and thus quantity that enters volume i either from left *or* right; module $\mathcal{R}$ decides on this based on the output of $\varphi_{\mathcal{A}}$ (which itself is a feedforward network, similarly to $\varphi_{\mathcal{D}}$). Technically, module $\mathcal{R}$ applies an upwind differencing scheme to prevent numerical instability in the first order spatial derivative calculation (Versteeg & Malalasekera, 1995) and is computed as

$$\mathcal{R}\left(\varphi_{\mathcal{A}}\left(u_i\right)\right) = \begin{cases} \mathrm{ReLU}\left(\varphi_{\mathcal{A}}\left(u_i\right)\right) & \text{on } f_{i-} \\ -\mathrm{ReLU}\left(-\varphi_{\mathcal{A}}\left(u_i\right)\right) & \text{on } f_{i+} \end{cases}. \tag{4}$$

It ensures that further processing of the advective flux from $\varphi_{\mathcal{A}}(u_i)$ is performed only on one control volume surface (either left or right), by switching on only the left surface element when $\varphi_{\mathcal{A}} > 0$ or only the right surface element when $\varphi_{\mathcal{A}} < 0$.

Finally, considering Eq. 4, the flux calculations turn into

$$f_{i-} = \varphi_{\mathcal{N}}(u_i, u_{i-1}) \cdot \Big(\varphi_{\mathcal{D}}(u_i) + \mathcal{R}\big(\varphi_{\mathcal{A}}(u_i)\big)\Big) \tag{5}$$

$$f_{i+} = \varphi_{\mathcal{N}}(u_i, u_{i+1}) \cdot \Big(\varphi_{\mathcal{D}}(u_i) + \mathcal{R}\big(\varphi_{\mathcal{A}}(u_i)\big)\Big) \tag{6}$$

$$\mathcal{F}_i = f_{i-} + f_{i+}. \tag{7}$$

Since the advective flux is only considered on one side of volume i (due to module $\mathcal{R}$), the summation of the numerical stencil from both f_{i-} and f_{i+} in Eq. 7 leads to $[-1, 1]$, being applied to $[u_i, u_{i-1}]$ when $\varphi_{\mathcal{A}} > 0$, or $[u_i, u_{i+1}]$ when $\varphi_{\mathcal{A}} < 0$ (i.e. only first order spatial derivative). For the diffusive flux calculation, on the other hand, the summation of the numerical stencil leads to the classical one-dimensional numerical Laplacian with $[1, -2, 1]$ applied to $[u_{i-1}, u_i, u_{i+1}]$, representing the second order spatial derivative, since the calculation of $\varphi_{\mathcal{D}}$ is performed on both surfaces (see subsection B.3 for a derivation). Altogether, module $\mathcal{R}$ generates the inductive bias to make $\varphi_{\mathcal{A}}$ only approximate the advective, and $\varphi_{\mathcal{D}}$ the diffusive flux.

Boundary Conditions A means of applying boundary conditions in a model is essential when solving PDEs. Currently available models mostly adopt convolution operations to model spatiotemporal processes. However, a convolution only allows a constant value to be padded at the domain boundaries (e.g. zero-padding or mirror-padding), which is only appropriate for the implementation of Dirichlet or periodic boundary condition types. Other types of frequently used boundary conditions are Neumann and Cauchy. These are defined as a derivative of the quantity of interest, and hence cannot be easily implemented in convolutional models. However, with certain pre-defined boundary condition types in FINN, the flux kernels at the boundaries are adjusted accordingly to allow for straightforward boundary condition consideration. For Dirichlet boundary condition, a constant value $u = u_b$ is set as the input u_{i-1} (for the flux kernel f_{i-}) or u_{i+1} (for f_{i+}) at the corresponding boundary. For Neumann boundary condition ν, the output of the flux kernel f_{i-} or f_{i+} at the corresponding boundary is set to be equal to ν. With Cauchy boundary condition, the solution-dependent derivative is calculated and set as u_{i-1} or u_{i+1} at the corresponding boundary.

State Kernel The state kernel $\mathcal{S}$ calculates the source/sink and storage terms of Eq. 1. The source/sink (if required) is learned using the module $\Phi_\psi(u) \approx q(u)$. The storage, $\frac{\partial u}{\partial t}$, is then calculated using the output of the flux kernel and module Φ_ψ of the state kernel:

$$\mathcal{S}_i = \mathcal{F}_i(u_{i-1}, u_i, u_{i+1}) + \Phi_\psi(u_i) \approx \frac{\partial u_i}{\partial t}. \tag{8}$$

By doing so, the PDE in Eq. 1 is now reduced to a system of coupled ordinary differential equations (ODEs), which are functions of u_{i-1}, u_i, u_{i+1}, and t. Thus, the solutions of the coupled ODE system can be computed via numerical integration over time. Since first order explicit approaches, such as the Euler method (Butcher, 2008), suffer from numerical instability (Courant et al., 1967; Isaacson & Keller, 1994), we employ the neural ordinary differential equation method (Neural ODE, Chen et al., 2018) to reduce numerical instability via the Runge-Kutta adaptive time-stepping strategy. The Neural ODE evaluates $\frac{\partial u_i}{\partial t}$ in form of FINN at an arbitrarily fine Δt and integrates FINN's output over time from t to $t+1$; the resulting $u_i^{(t+1)}$ is fed back into the network as input in the next time step, until $t = \tau$ (i.e. the last time step) is reached. Therefore, the entire training is performed in closed-loop, improving stability and accuracy of the prediction compared to networks trained with teacher forcing, i.e. one-step-ahead prediction (Praditia et al., 2020). The weight update is realized by applying backpropagation through time (indicated by red arrows in Figure 1). In short, FINN takes only the initial condition u at time $t = 0$ and propagates the dynamics forward interactively with Neural ODE.

4. Experiments, Results, and Discussion

4.1. Synthetic Dataset

To demonstrate FINN's performance in comparison to other models, four different equations are considered as applications. First, *Burgers' equation* (Basdevant et al., 1986) is chosen as a challenging function, as it is a non-linear PDE with $v(u) = u$ that could lead to a shock in the solution $u(x,t)$. Second, the *diffusion-sorption equation* (Nowak & Guthke, 2016) is selected with the non-linear retardation factor $R(u)$ as coefficient for the storage term, which contains a singularity $R(u) \to \infty$ for $u \to 0$ due to the parameter choice. Third, the two-dimensional Fitzhugh-Nagumo equation (Klaasen & Troy, 1984) as candidate for a *diffusion-reaction equation* (Turing, 1952) is selected, which is challenging because it consists of two non-linearly coupled PDEs to solve two main unknowns: the activator u_1 and the inhibitor u_2. Fourth, the Allen-Cahn equation with a cubic reaction term is chosen, leading to multiple jumps in the solution $u(x,t)$. Details on all four equations, data generation and architecture designs can be found in subsection C.1, subsection C.2, subsection C.3, subsection C.4 of the appendix, respectively.

For each problem, three different datasets are generated (by conventional numerical simulation): *train*, used to train the models, in-distribution test (*in-dis-test*), being the train data simulated with a longer time span to test the models' generalization ability (extrapolation), and out-of-distribution test (*out-dis-test*). *Out-dis-test* data are used to test a trained ML model under conditions that are far away from training conditions, not only in terms of querying outputs for unseen inputs. Instead, *out-dis-test* data query outputs with regards to changes *not* captured by the inputs. These are changes that the ML tool per definition cannot be made aware of during training. In this work, they are represented by data generated with different initial or boundary condition, to test the generalization ability of the models outside the training distributions. FINN is trained and compared with both spatiotemporal deep learning models such as TCN, ConvLSTM, DISTANA and physics-aware neural network models such as PINN and PhyDNet. All models are trained with ten different random seeds using PyTorch's default weight initialization, and mean and standard deviation of the prediction errors are summarized in Table 1 for *train*, *in-dis-test* and *out-dis-test*. Details of each run are reported in the appendix. It is noteworthy that PINN cannot be tested on the *out-dis-test* dataset, since PINN assumes that the unknown variable u is an explicit function of x and t, and hence, when the initial or boundary condition is changed, the function will also be different and no longer valid.

4.1.1. Results

Burgers' The predictions of the best trained model of each method for the *in-dis-test* and the *out-dis-test* data are shown in Figure 2 and Figure 3, respectively. Both TCN and ConvLSTM fail to produce reasonable predictions, but qualitatively the other models manage to capture the shape of the data sufficiently, even towards the end of the *in-dis-test* period, where closed loop prediction has been applied for 380 time steps (after 20 steps of teacher forcing, see subsection C.1 for details). DISTANA, PINN, and FINN stand out in particular, but FINN produces more consistent and accurate predictions, evidenced by the mean value of the prediction errors. When tested against data generated with a different initial condition (*out-dis-test*), all models except for TCN and PhyDNet perform well. However, FINN still outperforms the other models with a significantly lower prediction error. The advective velocity learned by FINN's module $\varphi_{\mathcal{A}}$ is shown in Figure 5 (top left) and verifies that it successfully learned the advective velocity to be described by an identity function.

Diffusion-Sorption The predictions of the best trained model of each method for the concentration u from the diffusion-sorption equation are shown in Figure 11 of the appendix. TCN and ConvLSTM are shown to perform poorly even on the *train* data, evidenced by the high mean value of the prediction errors. On *in-dis-test* data, all models successfully produce relatively accurate predictions. However, when tested against different boundary conditions (*out-dis-test*), only FINN is able to capture the modi-

Table 1: Comparison of MSE and standard deviation scores across ten repetitions between different deep learning (above dashed line) and physics-aware neural networks (below dashed line) on different equations. Best results reported in bold.

Equation	Model	Params	Dataset		
			Train	*In-dis-test*	*Out-dis-test*
Burgers' 1D	TCN	38 500	$(1.6 \pm 3.4) \times 10^{-1}$	$(1.8 \pm 3.1) \times 10^{-1}$	$(1.6 \pm 3.3) \times 10^{-1}$
	ConvLSTM	13 200	$(6.8 \pm 9.9) \times 10^{-2}$	$(1.2 \pm 1.1) \times 10^{-1}$	$(7.3 \pm 9.5) \times 10^{-2}$
	DISTANA	25 126	$(1.8 \pm 1.0) \times 10^{-4}$	$(4.0 \pm 3.1) \times 10^{-3}$	$(1.5 \pm 1.6) \times 10^{-3}$
	PINN	3 021	$(5.1 \pm 0.5) \times 10^{-4}$	$(5.0 \pm 8.4) \times 10^{-3}$	—
	PhyDNet	37 718	$(7.2 \pm 2.2) \times 10^{-5}$	$(1.8 \pm 1.6) \times 10^{-1}$	$(4.5 \pm 2.5) \times 10^{-2}$
	FINN	**421**	$\mathbf{(2.8 \pm 2.9) \times 10^{-6}}$	$\mathbf{(2.5 \pm 3.1) \times 10^{-6}}$	$\mathbf{(2.8 \pm 2.9) \times 10^{-6}}$
Diffusion-sorption 1D	TCN	3 834	$(9.7 \pm 13.5) \times 10^{-2}$	$(1.2 \pm 1.7) \times 10^{-1}$	$(1.1 \pm 1.4) \times 10^{-1}$
	ConvLSTM	3 960	$(3.2 \pm 2.9) \times 10^{-2}$	$(3.0 \pm 2.3) \times 10^{-2}$	$(5.8 \pm 4.0) \times 10^{-2}$
	DISTANA	3 739	$(4.6 \pm 2.5) \times 10^{-5}$	$(2.4 \pm 2.5) \times 10^{-3}$	$(4.6 \pm 4.6) \times 10^{-3}$
	PINN	3 042	$(4.7 \pm 8.4) \times 10^{-5}$	$(4.1 \pm 8.7) \times 10^{-3}$	—
	PhyDNet	37 815	$\mathbf{(3.5 \pm 1.7) \times 10^{-5}}$	$(9.1 \pm 15.4) \times 10^{-3}$	$(1.7 \pm 0.9) \times 10^{-2}$
	FINN	**528**	$(4.7 \pm 4.9) \times 10^{-5}$	$\mathbf{(1.3 \pm 1.3) \times 10^{-4}}$	$\mathbf{(4.1 \pm 4.0) \times 10^{-5}}$
Diffusion-reaction 2D	TCN	31 734	$(1.4 \pm 0.9) \times 10^{-2}$	$(4.7 \pm 2.1) \times 10^{-1}$	$(1.5 \pm 0.8) \times 10^{-1}$
	ConvLSTM	24 440	$(8.7 \pm 21.3) \times 10^{-3}$	$(9.3 \pm 4.9) \times 10^{-2}$	$(1.5 \pm 1.3) \times 10^{-2}$
	DISTANA	75 629	$(4.0 \pm 3.4) \times 10^{-3}$	$(1.8 \pm 0.6) \times 10^{-1}$	$(1.3 \pm 0.9) \times 10^{-2}$
	PINN	3 062	$(2.7 \pm 1.8) \times 10^{-4}$	$(7.0 \pm 6.3) \times 10^{-2}$	—
	PhyDNet	185 589	$\mathbf{(7.5 \pm 0.9) \times 10^{-5}}$	$(7.8 \pm 1.8) \times 10^{-2}$	$(3.5 \pm 1.3) \times 10^{-2}$
	FINN	**882**	$(1.3 \pm 0.3) \times 10^{-4}$	$\mathbf{(2.1 \pm 0.5) \times 10^{-3}}$	$\mathbf{(6.1 \pm 0.3) \times 10^{-3}}$
Allen-Cahn 1D	TCN	10 052	$(9.6 \pm 20.0) \times 10^{-2}$	$(2.2 \pm 2.8) \times 10^{-1}$	$(2.4 \pm 3.3) \times 10^{-1}$
	ConvLSTM	7 600	$(6.1 \pm 12.6) \times 10^{-2}$	$(3.3 \pm 2.3) \times 10^{-1}$	$(3.4 \pm 3.8) \times 10^{-1}$
	DISTANA	6 422	$(5.1 \pm 2.3) \times 10^{-4}$	$(4.5 \pm 1.7) \times 10^{-2}$	$(4.6 \pm 3.0) \times 10^{-2}$
	PINN	3 021	$(2.2 \pm 1.4) \times 10^{-5}$	$(4.0 \pm 6.5) \times 10^{-2}$	—
	PhyDNet	37 718	$(1.0 \pm 0.6) \times 10^{-4}$	$(5.8 \pm 2.0) \times 10^{-2}$	$(6.4 \pm 1.8) \times 10^{-1}$
	FINN	**422**	$\mathbf{(6.9 \pm 7.7) \times 10^{-6}}$	$\mathbf{(1.5 \pm 1.8) \times 10^{-5}}$	$\mathbf{(3.1 \pm 3.8) \times 10^{-6}}$

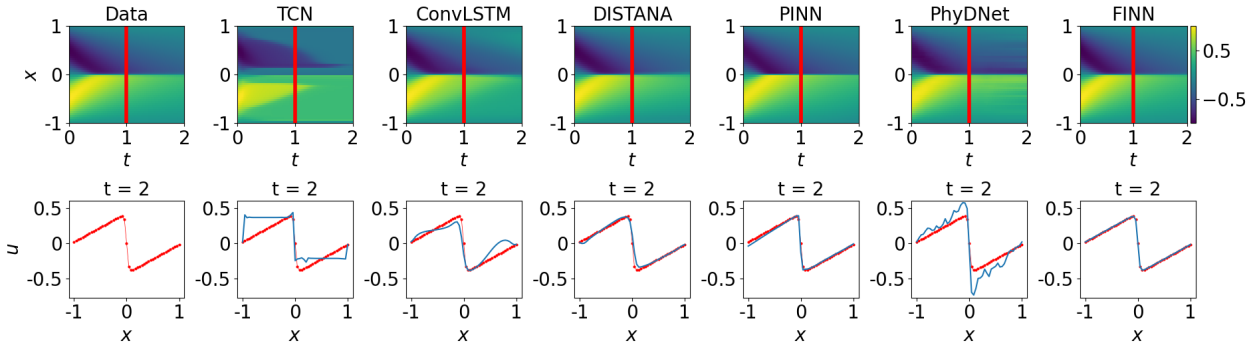


Figure 2: Plots of Burgers' data (red) and *in-dis-test* (blue) prediction using different models. The plots in the first row show the solution over x and t (the red lines mark the transition from *train* to *in-dis-test*), the second row visualizes the best model's solution distributed in x at $t = 2$.

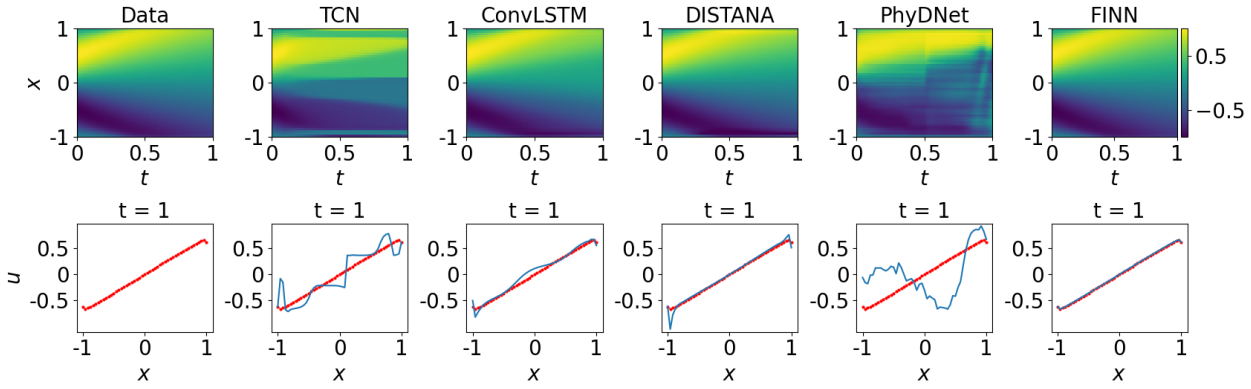


Figure 3: Plots of Burgers' data (red) and prediction (blue) of *out-dis-test* data using different models. The plots in the first row show the solution over x and t, the second row visualizes the best model's solution over x at $t = 1$.

fications and generalize well. The other models are shown to still overfit to the different boundary condition used in the train data (as detailed in subsection C.2). The retardation factor $R(u)$ learned by FINN's $\varphi_{\mathcal{D}}$ module is shown in Figure 5 (top center). The plot shows that the module learned the Freundlich retardation factor with reasonable accuracy.

Diffusion-Reaction The predictions of the best trained model of each method for activator u_1 in the diffusion-reaction equation are shown in Figure 14 (appendix) and Figure 4. TCN is again shown to fail to learn sufficiently from the *train* data. On *in-dis-test* data, DISTANA and the physics-aware models all predict with reasonable accuracy. When tested against data with different initial condition (*out-dis-test*), however, DISTANA and PhyDNet produce predictions with lower accuracy, and we find that FINN is the only model producing relatively low prediction errors. The reaction functions learned by FINN's Φ_{ψ} module are shown in Figure 5 (bottom). The plots show that the module successfully learned the Fitzhugh-Nagumo reaction function, both for the activator and inhibitor.

Allen-Cahn Results on the Allen-Cahn equation mostly align with those on Burgers' equation, confirming prior findings. However, it is worth noting that, again, only FINN is able to clearly represent the multiple nonlinearities caused by the exponent of third order in the reaction term, as visualized in Figure 15 and Figure 16 of the appendix. Again, Figure 5 (top right) shows that FINN accurately learned the dataset's reaction function.

4.1.2. Discussion

Overall, even with a high number of parameters, the prediction errors obtained using the pure ML methods (TCN, ConvLSTM, DISTANA) are worse compared to the physics-aware models, as shown in Table 1. As a physics-aware model, PhyDNet also possesses a high number of parameters. However, most of the parameters are allocated to the data-driven part (i.e. ConvLSTM branch), compared to the physics-aware part (i.e. PhyCell branch). In contrast to the other pure ML methods, DISTANA predicts with higher accuracy. This could act as an evidence that appropriate structural design of a neural network is as essential as physical knowledge to regularize the model training.

Among the physics-aware models, PINN and PhyDNet lie on different extremes. On one side, PINN requires complete knowledge of the modelled system in form of the equation. This means that all the functions, such as the advective velocity in Burgers' equation, the retardation factor in the diffusion-sorption equation, and the reaction functions in the diffusion-reaction and Allen-Cahn equations have to be pre-defined in order to train the model. This leaves less room for learning from data and could be error-prone if the designer's assumption is incorrect. On the other side, PhyDNet relies more heavily on the data-driven part and, therefore, could overfit the train data. This can be shown by the fact that PhyDNet reaches the lowest training errors for the diffusion-sorption and diffusion-reaction equation predictions compared to the other models, but its performance significantly deteriorates when applied to *in-* and *out-dis-test* data. Our proposed model, FINN, lies somewhere in these two extremes, compromising between putting sufficient physical knowledge into the model and leaving room for learning from data. As a consequence, we observe FINN showing excellent generalization ability. It significantly outperforms the other models up to multiple orders of magnitude, especially on *out-dis-test* data, when tested with different initial and boundary conditions; which is considered a particularly challenging task for ML models. Furthermore, the structure of FINN allows the extractions of learned functions such as the advective velocity,

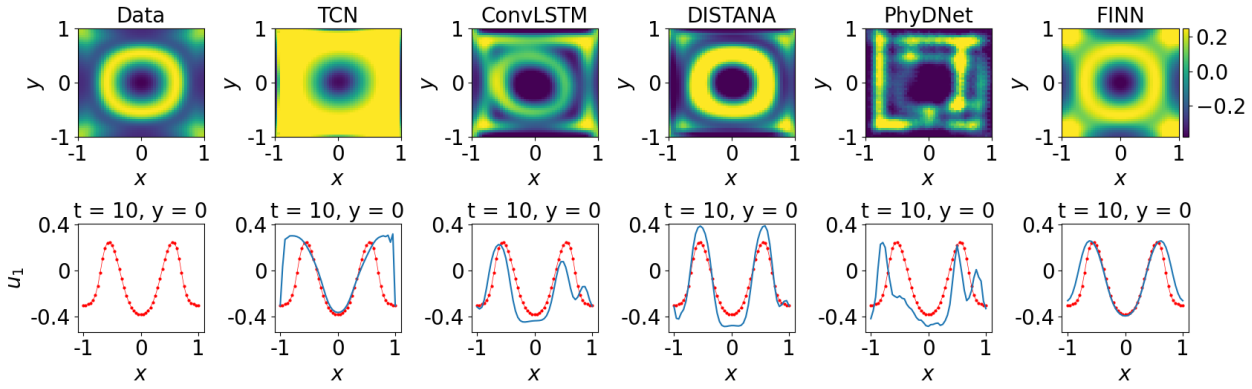


Figure 4: Plots of diffusion-reaction's activator data (red) and prediction (blue) of unseen dataset using different models. The plots in the first row show the solution distributed over x and y at $t = 10$, and the plots in the second row show the solution distributed in x at $y = 0$ and $t = 10$.

retardation factor, and reaction functions, showing good interpretability of the model.

We also show that FINN properly handles the provided numerical boundary condition, as evidenced when applied to the test data that is generated with a different left boundary condition value, visualized in Figure 11 (appendix). Here, the test data is generated with a Dirichlet boundary condition $u(0, t) = 0.7$, which is lower than the value used in the train data, $u(0, t) = 1.0$. However, FINN is the only model that appropriately processes this different boundary condition value so that the prediction fits the test data nicely. The other models overestimate their prediction by consistently showing a tendency to still fit the prediction to the higher boundary condition value encountered during training.

Even though the spatial resolution used for the synthetic data generation is relatively coarse, leading to sparse data

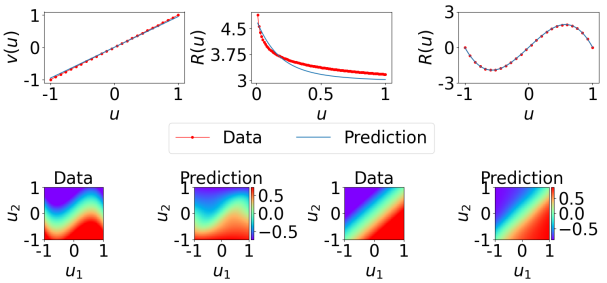


Figure 5: Plots of the learned functions (blue) as a function of u compared to the data (red) for Burgers' (top left), diffusion-sorption (top center), and Allen-Cahn (top right). The learned activator u_1 and inhibitor u_2 reaction functions in the diffusion-reaction equation are contrasted to the corresponding ground truth ('Prediction' and 'Data' plots in the second row).

availability, FINN generalizes well. PINN, on the other hand, slightly suffers from the low resolution of the train data, although it still shows reasonable performance for the three test cases. Nevertheless, we conducted an experiment showing that PINN performs slightly better and more consistently when trained on higher resolution data (see appendix, subsection D.4), albeit still worse than FINN on coarse data. Therefore, we conclude that FINN is also applicable to real observation data that are often available only in low resolution, and/or in limited quantity. We demonstrate this further in subsection 4.2, when we apply FINN to real experimental data. FINN's generalization ability is superior to PINN, due to the fact that it is not possible to apply a trained PINN model to predict data with different initial or boundary condition.

In terms of interpretability, FINN allows the extraction of functions learned by its dedicated modules. These learned functions can be compared with the real functions that generated the data (at least in the synthetic data case); examples are shown in Figure 5. The learned functions are the main data-driven discovery part of FINN and can also be used as a physical plausibility check. PhyDNet also comprises of a physics-aware and a data-driven part. However, it is difficult, if not impossible, to infer the learned equation from the model. Furthermore, the data-driven part of PhyDNet does not possess comparable interpretability, and could lead to overfitting as discussed earlier.

4.1.3. Ablations

In a number of ablation studies, we shed light on the relevance of particular choices in FINN. As such, we substantiate the choice of feedforward modules over polynomial regression in subsection D.1, proof FINN's ability to adequately learn unknown constituents to be robust to noise in

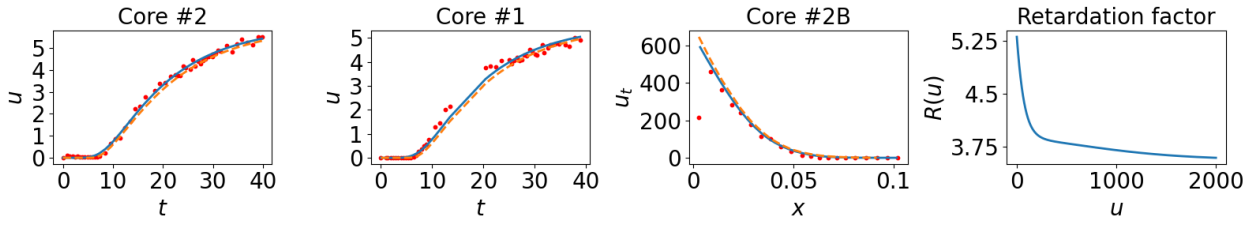


Figure 6: Breakthrough curve prediction of FINN (blue line) during training using data from core sample #2 (left), during testing using data from core sample #1 (second from left) and total concentration profile prediction using data from core sample #2B (second from right). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The right-most plot shows the learned retardation factor $R(u)$.

subsection D.2, and consolidate the advantages of applying Neural ODE compared to traditional closed-loop application and Euler integration in subsection D.3.

4.2. Experimental Dataset

Real observation data are often available in limited amount, and they only provide partial information about the system of interest. To demonstrate how FINN can cope with real observation data, we use experimental data for the diffusion-sorption problem. The experimental data are collected from three different core samples that are taken from the same geographical area: #1,#2, and #2B (see subsection C.5 in the appendix for details). The objective of this experiment is to learn the retardation factor function from one of the core samples that concurrently applies to all the other samples. For this particular purpose, we only implement FINN since the other models have no means of learning the retardation factor explicitly. Here, we use the module $\varphi_{\mathcal{D}}$ to learn the retardation factor function, and we assume that the diffusion coefficient values of all the core samples are known. FINN is trained with the breakthrough curve of u, which is the dissolved concentration only at $x = L|_{0 \leq t \leq t_{\text{end}}}$ (i.e. only 55 data points).

Results and Discussion FINN reaches a higher accuracy for the training with core sample #2, with $\text{MSE} = 4.84 \times 10^{-4}$ compared to a calibrated physical model from Nowak & Guthke (2016) with $\text{MSE} = 1.06 \times 10^{-3}$, because the latter has to assume a specific function $R(u)$ with a few parameters for calibration. Our learned retardation factor is then applied and tested to core samples #1 and #2B. Figure 6 shows that FINN's prediction accuracy is competitive compared to the calibrated physical model. For core sample #1, FINN's prediction has an accuracy of 1.37×10^{-3} compared to the physical model that underestimates the breakthough curve (i.e. concentration profile) with $\text{MSE} = 2.50 \times 10^{-3}$. Core sample #2B has significantly longer length than the other samples, and therefore a no-flow Neumann boundary condition was assumed at the bottom of the core. Because there is no breakthrough curve data available for this specific sample, we compare the prediction against the so-called total concentration profile $u(x, t_{\text{end}})$ at the end of the experiment. FINN produced a prediction with an accuracy of 1.16×10^{-3}, whereas the physical model overestimates the concentration with $\text{MSE} = 2.73 \times 10^{-3}$. To briefly summarize, FINN is able to learn the retardation factor from a sparse experimental dataset and apply it to other core samples with similar soil properties with reasonable accuracy, even when a different boundary condition type is applied.

5. Conclusion

Spatiotemporal dynamics often can be described by means of advection-diffusion type equations, such as Burgers', diffusion-sorption, or diffusion-reaction equations. When modeling those dynamics with ANNs, large efforts must be taken to prevent the models from overfitting (given the model is able to learn the problem at all). The incorporation of physical knowledge as regularization yields robust predictions beyond training data distributions.

With FINN, we have introduced a modular, physics-aware neural network with excellent generalization abilities beyond different initial and boundary conditions, when contrasted to pure ML models and other physics-aware models. FINN is able to model and extract unknown constituents of differential equations, allowing high interpretability and an assessment of the plausibility of the model's out-of-distribution predictions. As next steps we seek to apply FINN beyond second order spatial derivatives, improve its scalability to large datasets, and make it applicable to heterogeneously distributed data (i.e. represented as graphs) by modifying the module $\varphi_{\mathcal{N}}$ to approximate variable and location-specific stencils (for more details on limitations of FINN, the reader is referred to subsection B.2). Another promising future direction is the application of FINN to real-world weather data.

Software and Data

Code and data that are used for this paper can be found in the repository ***** *[hidden for anonymization reasons. Instead, the supplementary material of this submission contains an anonymous version of the repository's* `README.md` *file along with the according data and model scripts.]*

Acknowledgements

Hidden for anonymization reasons

References

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Basdevant, C., Deville, M., Haldenwang, P., Lacroix, J., Ouazzani, J., Peyret, R., Orlandi, P., and Patera, A. Spectral and finite difference solutions of the burgers equation. *Computers & Fluids*, 14(1):23–41, 1986. doi: https://doi.org/10.1016/0045-7930(86)90036-8.

Battaglia, P., Hamrick, J., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Butcher, J. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008. ISBN 9780470753750.

Chen, R., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Courant, R., Friedrichs, K., and Lewy, H. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967. doi: 10.1147/rd.112.0215.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Fornberg, B. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

Guen, V. and Thome, N. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11474–11484, 2020.

Isaacson, E. and Keller, H. *Analysis of Numerical Methods*. Dover Books on Mathematics. Dover Publications, 1994. ISBN 9780486680293.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A., Graves, A., and Kavukcuoglu, K. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Karlbauer, M., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., and Butz, M. A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint arXiv:1912.11141*, 2019.

Klaasen, G. and Troy, W. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1):96–110, 1984. doi: 10.1137/0144008.

Kochkov, D., Smith, J., Alieva, A., Wang, Q., Brenner, M., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

Lake, B. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems 32*, pp. 9791–9801. 2019.

Lake, B., Ullman, T., Tenenbaum, J., and Gershman, S. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 2017. doi: 10.1017/S0140525X16001837.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Long, Z., Lu, Y., Ma, X., and Dong, B. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pp. 3208–3216. PMLR, 2018.

Moukalled, F., Mangani, L., and Darwish, M. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. doi: 10.1007/978-3-319-16874-6.

Nowak, W. and Guthke, A. Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, 18(11), 2016. doi: 10.3390/e18110409.

Praditia, T., Walser, T., Oladyshkin, S., and Nowak, W. Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture. *Energies*, 13(15):3873, 2020. doi: 10.3390/en13153873.

Praditia, T., Karlbauer, M., Otte, S., Oladyshkin, S., Butz, M., and Nowak, W. Finite volume neural network: Modeling subsurface contaminant transport. *arXiv preprint arXiv:2104.06010*, 2021.

Raissi, M., Perdikaris, P., and Karniadakis, G. 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. doi: 10.1016/j.jcp.2018.10.045.

Riley, K., Hobson, M., and Bence, S. *Mathematical Methods for Physics and Engineering*. Cambridge University Press, 2009. ISBN 9780521139878.

Seo, S., Meng, C., and Liu, Y. Physics-aware difference graph networks for sparsely-observed dynamics. In *International Conference on Learning Representations*, 2019.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*, 2015.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

Smolarkiewicz, P. A simple positive definite advection scheme with small implicit diffusion. *Monthly Weather Review*, 111(3):479 – 486, 1983. doi: 10.1175/1520-0493.

Turing, A. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

Versteeg, H. and Malalasekera, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. New York, 1995. ISBN 9780470235157.

Yin, Y., Guen, V., Dona, J., Ayed, I., de Bézenac, E., Thome, N., and Gallinari, P. Augmenting physical models with deep networks for complex dynamics forecasting. *arXiv preprint arXiv:2010.04456*, 2020.

Zhuang, J., Kochkov, D., Bar-Sinai, Y., Brenner, M., and Hoyer, S. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6(6):064605, 2021.

A. Appendix

This appendix is structured as follows: Additional detailed differences between FINN and the benchmark models used in this work are presented in subsection B.1. Then, limitations of FINN are discussed briefly in subsection B.2. Detailed numerical derivation of the first and second order spatial derivative is shown in subsection B.3. Additional details on the equations used as well as model specifications and in-depth results are presented in subsection C.1 (Burgers'), subsection C.2 (diffusion-sorption), subsection C.3 (diffusion-reaction), and subsection C.4 (Allen-Cahn). Moreover, the soil parameters and simulation domain used in the diffusion-sorption laboratory experiment are presented in subsection C.5. The results of our ablation studies are reported in Appendix D. Additional polynomial-fitting baselines to account for the equations are outlined in subsection D.1, including an ablation where we replace the neural network modules of FINN by polynomials. A robustness analysis of FINN can be found in subsection D.2, where our method is evaluated on noisy data from all equations. The role of the Neural ODE module is evaluated in subsection D.3, accompanied with a runtime analysis. Finally, results of training PINN with high-resolution data and training PhyDNet with the original architecture (more parameters) are reported in subsection D.4 and subsection D.5, respectively.

B. Methodological Supplements

B.1. Distinction to Related Work

Pure ML models Originally, FINN was inspired by DISTANA, a pure ML model proposed by Karlbauer et al. (2019). While DISTANA has large similarities to Shi et al. (2015)'s ConvLSTM, it propagates lateral information through an additional latent map—instead of reading lateral information from the input map directly, as done in ConvLSTM—and transforms that lateral information by means of a user-defined combination of arbitrary layers. Accordingly, the processing of the two-point flux approximation in FINN using the lateral information, is more akin to DISTANA, albeit motivated and augmented by physical knowledge.

As a result, the lateral information flow is guaranteed to behave in a physically plausible manner. That is, quantity can either be locally generated by a source (increased), locally absorbed by a sink (decreased), or spatially distributed (move to neighboring cells). Terminology separates the spatially distribution of quantity into diffusion and advection. Diffusion describes the equalization of quantity from high to low concentration levels, whereas advection is defined as the bulk motion of a large group of particles/atoms caused by external forces. FINN ensures the

conservation of laterally propagating quantity, i.e. what flows from left to right will effectively cause a decrease left and an increase right. Pure ML models (without physical constraints) cannot be guaranteed to adhere to these fundamental rules.

PINN Since ML models guided by physical knowledge not only behave empirically plausible but also require fewer parameters and generalize to much broader extent, numerous approaches have been proposed recently to combine artificial neural networks with physical knowledge. Raissi et al. (2019) introduced the physics-informed neural network (PINN), a concrete and outstanding model that explicitly learns a provided PDE. As a result, PINN mimics e.g. Burgers' equation (see equation 15) by learning the quantity function $u(x,t)$ for defined initial and boundary conditions with a feedforward network. The partial derivatives are implicitly realized by automatic differentiation of the feedforward network (representing u) with respect to the input variable x or t. The learned neural network thus satisfies the constraints defined by the partial derivatives.

This method has the advantage that it explicitly provides a solution of the desired function $u(x,t)$ and, correspondingly, predictions can be generated for an arbitrary combination of input values, circumventing the need for simulating the entire domain with e.g. a carefully chosen simulation step size. In contrast, FINN does not learn the explicit function $u(x,t)$ defined for particular initial and boundary conditions, but approximates the distinct components of the function. These components are combined as suggested by the physical equation to result in a compositional model that is more universally applicable. That is, in stark contrast to PINN, the compositional function learned by FINN can be applied to varying initial and boundary conditions, since the learned individual components provide the same functionality as the corresponding components of the PDE when processed by a numerical solver. Applying the numerical solver, i.e. the finite volume method (FVM), however, requires either complete knowledge of the equation or careful calibration of the unknowns by choosing equations from a library of possible solutions and tuning the parameters. FINN's data driven component can reveal unknown relations, such as the retardation factor of a function, without the need for subjective prior assumptions.

Learning Derivatives via Convolution An alternative and much addressed approach of implanting physical plausibilty into artificial neural networks is to implicitly learn the first n derivatives using appropriately designed convolution kernels (Long et al., 2018; Li et al., 2020a;b; Guen & Thome, 2020; Yin et al., 2020; Sitzmann et al., 2020). These methods exploit the link that most PDEs, such as $u(t,x,y,\dots)$, can be reformulated as

$$\frac{\partial u}{\partial t} = F\left(t, x, y, \dots, u, \frac{\partial u}{\partial x}, \frac{\partial u}{y}, \frac{\partial u}{\partial x \partial y}, \frac{\partial^2 u}{\partial x^2}, \dots\right). \tag{9}$$

When learning partial derivatives up to order n and setting irrelevant features to zero, these methods have, in principle, the capacity to represent most PDEs. However, the degrees of freedom in these methods are still very high and can fail to safely guide the ML algorithm. FINN accounts for the first and second derivative by learning an according stencil in the module $\varphi_{\mathcal{N}}$ and combining this stencil with the case-sensitive ReLU module $\mathcal{R}$, which allows a precise control of the information flow resulting from the first and second derivative. More importantly, convolutional structure only allows implementation of Dirichlet and periodic boundary condition (by means of zero- or mirror-padding), and is not appropriate for implementation of other boundary condition types.

Learning ODE Coefficients The group around Brenner and Hoyer (Bar-Sinai et al., 2019; Kochkov et al., 2021; Zhuang et al., 2021) follow another line of research about learning the coefficients of ordinary differential equations (ODEs); note that PDEs can be transformed into a set of coupled ODEs by means of polynomial expansion or spectral differentiation as shown in Bar-Sinai et al. (2019). The physical constraint in these works is mostly realized by satisfying the temporal derivative as loss.

While this approach shares similarities with our method by incorporating physically motivated inductive bias into the model, it uses this bias mainly to improve interpolation from coarse to finer grid resolutions and thus to accelerate simulation. Our work focuses on discovering unknown relationships/laws (or re-discovering laws in the case of the synthetic examples), such as the advective velocity in the Burgers' example, the retardation factor function in the diffusion-sorption example, and the reaction function in the diffusion-reaction example. Additionally, the works by Brenner and Hoyer employ a convolutional structure, which is only applicable to Dirichlet or periodic boundary conditions, and it suffers from a slight instability during training when the training data trajectory is unrolled for a longer period. In contrast, FINN employs the flux kernel, calculated at all control volume surfaces, which enables the implementation and discovery of various boundary conditions. Furthermore and in contrast to Brenner and Hoyer, FINN employs the Neural ODE method as the time integrator to reduce numerical instability during training with long time series. However and in accord with this line of research, FINN is also able to generalize well when trained with a relatively sparse dataset (coarse resolution), reducing the computational burden.

Numerical ODE Solvers Traditional numerical solvers for ordinary differential equations (ODEs) can be seen as the pure-physics contrast to FINN. However, these do not have a learning capacity to reveal unknown dependencies or functions from data. Also, as shown in (Yin et al., 2020), a pure Neural ODE without physical inductive bias does not reach the same level of accuracy as physics-aware neural network models. This is underlined by our field experiment where FINN reached lower errors on a real-world dataset compared to a conventional FVM model.

B.2. Limitations of FINN

While the largest limitation of our current method can be seen in the capacity to only represent first and second order spatial derivatives, this is an issue that we will address in follow up work. Still, FINN can already be applied to a very wide range of problems as most equations in fact only depend on up to the second spatial derivative. So far, FINN is only applicable to spatially homogeneously distributed data—we intend to extend it to heterogeneous data from graphs. Although we have successfully applied FINN to 2D diffusion reaction data, the training time is considerable. While, according to our observations, this appears to be a common issue for physics-aware neural networks, the implementation of custom-convolution layers could widen this bottleneck with today's hardware-accelerated computation of convolution operations.

B.3. Learning the Numerical Stencil

Semantically, the $\varphi_{\mathcal{N}}$ module learns the numerical stencil, that is the geometrical arrangement of a group of neighboring cells to approximate the derivative numerically, effectively learning the first spatial derivative $\frac{\partial u}{\partial x}$ from both $[u_{i-1}, u_i]$ and $[u_i, u_{i+1}]$, which are the inputs to the $\varphi_{\mathcal{N}_{i-}}$ and $\varphi_{\mathcal{N}_{i+}}$ module, respectively.

The lateral information flowing from u_{i-1} and u_{i+1} toward u_i is controlled by the $\varphi_{\mathcal{A}}$ (advective flux, i.e. bulk motion of many particles/atoms that can either move to the left or to the right) and the $\varphi_{\mathcal{D}}$ (diffusive flux, i.e. drive of particles/atoms to equilibrium from regions of high to low concentration) modules. Since the advective flux can only move either to the left or to the right, it will be considered only in the left flux kernel (f_{i-}) or in the right flux kernel (f_{i+}), and not both at the same time. The case-sensitive ReLU module $\mathcal{R}$ (Eq. 4) decides on this, by setting the advective flux in the irrelevant flux kernel to zero (effectively depending on the sign of the output of $\varphi_{\mathcal{A}}$). Thus, the advective flux is only considered from either u_{i-1} or u_{i+1} to u_i, which amounts to the first order spatial derivative.

The diffusive flux, on the other hand, can propagate from both sides towards the control volume of interest u_i and, hence, the second order spatial derivative, accounting for the difference between u_{i-1} and u_{i+1}, has to be applied. In our method, this is realized through the combination of the $\varphi_{\mathcal{N}}$ and $\varphi_{\mathcal{D}}$ modules, calculating the diffusive fluxes $\delta_- = \varphi_{\mathcal{N}}(u_{i-1}, u_i)\varphi_{\mathcal{D}}(u_i)$ and $\delta_+ = \varphi_{\mathcal{N}}(u_i, u_{i+1})\varphi_{\mathcal{D}}(u_i)$ inside of the respective left and right flux kernel. The combination of these two deltas in the state kernel ensures the consideration of the diffusive fluxes from left (including u_{i-1}) and from right (including u_{i+1}), resulting in the ability to account for the second order spatial derivative.

Technically, the first, i.e. $[-1, 1]$, and second, i.e. $[1, -2, 1]$, order spatial differentiation schemes are common definitions and a derivation can be found, for example, in Fornberg (1988). However, a quick derivation of the Laplace scheme $[1, -2, 1]$ can be formulated as follows. Define the second order spatial derivative as the difference between two first order spatial derivatives, i.e.

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{(\partial u/\partial x)|_{i-} - (\partial u/\partial x)|_{i+}}{\Delta x} \tag{10}$$

with the two first order spatial derivatives—representing the differences between u_{i-1}, u_i (left, i.e. 'minus') and u_i, u_{i+1} (right, i.e. 'plus')—defined as

$$(\partial u/\partial x)|_{i-} \approx (u_{i-1} - u_i)/\Delta x \tag{11}$$

$$(\partial u/\partial x)|_{i+} \approx (u_i - u_{i+1})/\Delta x. \tag{12}$$

Then, substituting Eq. 11 and Eq. 12 into Eq. 10, we get

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{(u_{i-1} - u_i) - (u_i - u_{i+1})}{\Delta x^2} \tag{13}$$

$$\approx \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}, \tag{14}$$

hence the $[+1, -2, +1]$ as coefficients in the second order spatial derivative.

C. Data and Model Details

C.1. Burgers'

The Burgers' equation is commonly employed in various research areas, including fluid mechanics.

Data The 1D generalized Burgers' equation is written as

$$\frac{\partial u}{\partial t} = -v(u)\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2}, \tag{15}$$

where the main unknown is u, the advective velocity is denoted as $v(u)$ which is a function of u and the diffusion coefficient is $D = 0.01/\pi$. In the current work, the advective velocity function is chosen to be an identity function $v(u) = u$ to reproduce the experiment conducted in the PINN paper (Raissi et al., 2019).

Table 2: Closed loop MSE on the *train* data from ten different training runs for each model for the Burgers' equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	4.1×10^{-2}	7.9×10^{-2}	1.2×10^{-4}	6.1×10^{-4}	8.0×10^{-5}	1.9×10^{-6}
2	5.1×10^{-2}	1.7×10^{-1}	1.8×10^{-4}	4.1×10^{-4}	4.7×10^{-5}	1.5×10^{-6}
3	3.6×10^{-2}	1.3×10^{-2}	6.4×10^{-5}	4.9×10^{-4}	6.8×10^{-5}	2.2×10^{-6}
4	4.3×10^{-2}	2.7×10^{-4}	2.9×10^{-4}	5.6×10^{-4}	1.2×10^{-4}	2.5×10^{-6}
5	5.4×10^{-2}	2.1×10^{-4}	2.2×10^{-4}	5.1×10^{-4}	6.7×10^{-5}	1.6×10^{-6}
6	3.9×10^{-2}	1.5×10^{-3}	1.3×10^{-4}	5.2×10^{-4}	5.5×10^{-5}	9.9×10^{-7}
7	6.6×10^{-2}	4.1×10^{-4}	5.4×10^{-5}	5.2×10^{-4}	5.1×10^{-5}	2.2×10^{-6}
8	1.2×10^{0}	3.6×10^{-2}	2.7×10^{-4}	4.8×10^{-4}	5.6×10^{-5}	3.0×10^{-6}
9	3.4×10^{-2}	5.5×10^{-2}	3.9×10^{-4}	5.5×10^{-4}	9.9×10^{-5}	1.1×10^{-5}
10	6.8×10^{-2}	3.2×10^{-1}	1.1×10^{-4}	5.0×10^{-4}	8.1×10^{-5}	4.5×10^{-7}

Table 3: Closed loop MSE on *in-dis-test* data from ten different training runs for each model for the Burgers' equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	3.0×10^{-2}	1.2×10^{-1}	2.6×10^{-3}	4.1×10^{-3}	1.1×10^{-1}	1.6×10^{-6}
2	7.0×10^{-2}	3.7×10^{-1}	1.1×10^{-3}	1.3×10^{-3}	2.5×10^{-1}	1.3×10^{-6}
3	2.4×10^{-2}	2.0×10^{-1}	8.9×10^{-4}	2.9×10^{-3}	3.4×10^{-2}	1.9×10^{-6}
4	5.1×10^{-2}	5.6×10^{-3}	6.2×10^{-3}	3.7×10^{-3}	4.7×10^{-1}	2.1×10^{-6}
5	5.4×10^{-2}	2.6×10^{-3}	2.6×10^{-3}	2.4×10^{-3}	1.1×10^{-2}	1.3×10^{-6}
6	3.2×10^{-2}	1.0×10^{-2}	1.2×10^{-2}	1.9×10^{-3}	3.2×10^{-1}	6.8×10^{-7}
7	8.8×10^{-2}	5.5×10^{-3}	1.1×10^{-3}	2.1×10^{-3}	3.3×10^{-1}	1.8×10^{-6}
8	1.1×10^{0}	1.6×10^{-1}	4.0×10^{-3}	4.0×10^{-4}	2.7×10^{-1}	2.4×10^{-6}
9	3.3×10^{-2}	1.9×10^{-1}	4.0×10^{-3}	3.0×10^{-2}	2.6×10^{-2}	1.2×10^{-5}
10	3.6×10^{-1}	9.8×10^{-2}	6.0×10^{-3}	1.5×10^{-3}	8.0×10^{-3}	2.8×10^{-7}

Table 4: Closed loop MSE on *out-dis-test* data from ten different training runs for each model for the Burgers' equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	3.6×10^{-2}	7.2×10^{-2}	5.9×10^{-3}	-	5.3×10^{-2}	1.9×10^{-6}
2	3.4×10^{-2}	1.9×10^{-1}	4.4×10^{-4}	-	2.1×10^{-2}	1.5×10^{-6}
3	3.3×10^{-2}	2.9×10^{-3}	2.9×10^{-3}	-	4.5×10^{-2}	2.2×10^{-6}
4	3.7×10^{-2}	8.9×10^{-4}	1.4×10^{-3}	-	2.0×10^{-2}	2.5×10^{-6}
5	4.2×10^{-2}	1.2×10^{-3}	1.0×10^{-3}	-	3.1×10^{-2}	1.6×10^{-6}
6	2.7×10^{-2}	1.4×10^{-3}	3.0×10^{-4}	-	1.1×10^{-1}	9.9×10^{-7}
7	6.7×10^{-2}	3.3×10^{-2}	6.2×10^{-4}	-	2.8×10^{-2}	2.2×10^{-6}
8	1.1×10^{0}	4.0×10^{-2}	7.0×10^{-4}	-	4.7×10^{-2}	3.0×10^{-6}
9	3.0×10^{-2}	8.5×10^{-2}	7.4×10^{-4}	-	4.5×10^{-2}	1.1×10^{-5}
10	1.2×10^{-1}	3.1×10^{-1}	7.0×10^{-4}	-	4.5×10^{-2}	4.5×10^{-7}

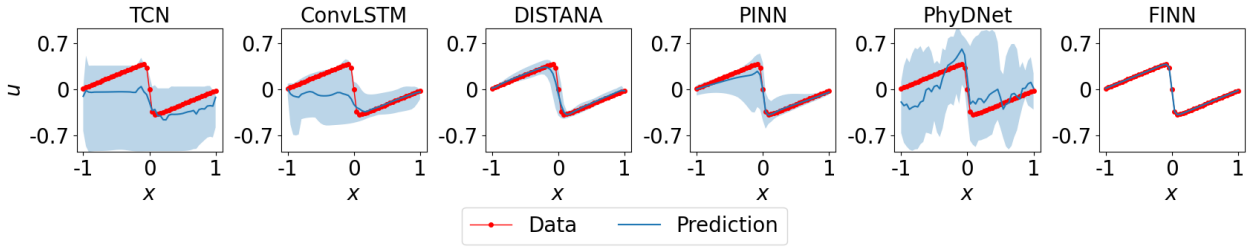


Figure 7: Prediction mean over ten different trained models (with 95% confidence interval) of the Burgers' equation at $t = 2$ for the *in-dis-test* dataset.

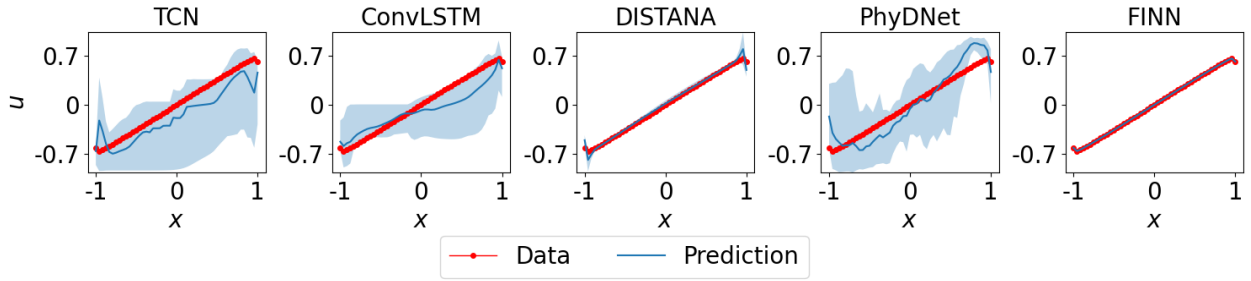


Figure 8: Prediction mean over ten different trained models (with 95% confidence interval) of the Burgers' equation at $t = 1$ for the *out-dis-test* data.

The simulation domain for the ***train*** **data** is defined with $x = [-1, 1]$, $t = [0, 1]$ and is discretized with $N_x = 49$ spatial locations, and $N_t = 201$ simulation steps. The initial condition is defined as $u(x, 0) = -\sin(\pi x)$, and the boundary condition is defined as $u(-1, t) = u(1, t) = 0$.

In-dis-test **data** is simulated with $x = [-1, 1]$ and a time span of $t = [1, 2]$ and $N_t = 401$. Initial condition is taken from the train data at $t = 1$ and boundary conditions are also similar to the train data.

The simulation domain for the ***out-dis-test*** **data** is identical with the train data, except for the initial condition that is defined as $u(x, 0) = \sin(\pi x)$.

Model Architectures Both **TCN** and **ConvLSTM** are designed to have one input neuron, one hidden layer of size 32, and one output neuron. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and a hidden layer of size 32 is used. The pure ML models were trained on the first 150 time steps and validated on the remaining 50 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done during teacher forcing. **PINN** was defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 1] (8 hidden layers, each contains 20 hidden neurons), as reported in the original work by Raissi et al. (2019). **PhyDNet** was defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer. For **FINN**, the modules $\varphi_{\mathcal{N}}$, $\varphi_{\mathcal{D}}$, $\mathcal{R}$ and $\varphi_{\mathcal{A}}$ were used, with $\varphi_{\mathcal{A}}$ defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes u as an input and outputs the advective velocity $v(u)$, and $\varphi_{\mathcal{D}}$ as a learnable scalar that learns the diffusion coefficient D. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 2), *in-dis-test* (Table 3 and Figure 7), and *out-dis-test* (Table 4 and Figure 8) datasets.

C.2. Diffusion-Sorption

The diffusion-sorption equation is another widely applied equation in fluid mechanics. A practical example of the equation is to model contaminant transport in groundwater. Its retardation factor R can be modelled using different closed parametric relations known as sorption isotherms that should be calibrated to observation data.

Table 5: Closed loop MSE on the *train* data from ten different training runs for each model for the diffusion-sorption equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	3.6×10^{-2}	4.8×10^{-5}	2.1×10^{-5}	2.2×10^{-4}	3.1×10^{-5}	1.1×10^{-4}
2	2.0×10^{-1}	2.3×10^{-2}	1.6×10^{-5}	2.3×10^{-6}	2.5×10^{-5}	1.4×10^{-5}
3	8.1×10^{-2}	4.4×10^{-2}	2.7×10^{-5}	1.2×10^{-6}	3.2×10^{-5}	1.1×10^{-4}
4	2.0×10^{-1}	1.0×10^{-2}	9.3×10^{-5}	2.1×10^{-4}	2.9×10^{-5}	4.1×10^{-5}
5	1.8×10^{-3}	8.7×10^{-2}	6.2×10^{-5}	2.5×10^{-6}	7.7×10^{-5}	5.3×10^{-6}
6	2.7×10^{-4}	5.9×10^{-2}	4.0×10^{-5}	1.7×10^{-6}	4.4×10^{-5}	1.9×10^{-5}
7	3.2×10^{-4}	6.4×10^{-2}	7.7×10^{-5}	8.6×10^{-6}	1.8×10^{-5}	1.5×10^{-5}
8	1.7×10^{-2}	1.4×10^{-4}	2.5×10^{-5}	2.0×10^{-6}	2.4×10^{-5}	1.2×10^{-4}
9	4.3×10^{-1}	2.2×10^{-3}	5.9×10^{-5}	9.0×10^{-6}	2.1×10^{-5}	1.2×10^{-5}
10	6.6×10^{-4}	2.7×10^{-2}	3.9×10^{-5}	1.1×10^{-5}	5.2×10^{-5}	1.7×10^{-5}

Table 6: Closed loop MSE on *in-dis-test* data from ten different training runs for each model for the diffusion-sorption equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.9×10^{-2}	7.7×10^{-3}	6.9×10^{-4}	3.0×10^{-2}	3.8×10^{-3}	3.2×10^{-4}
2	2.4×10^{-1}	1.0×10^{-2}	8.5×10^{-5}	8.6×10^{-4}	8.4×10^{-3}	3.5×10^{-5}
3	4.7×10^{-2}	2.1×10^{-2}	1.7×10^{-3}	9.3×10^{-5}	2.9×10^{-4}	3.2×10^{-4}
4	2.4×10^{-1}	2.8×10^{-2}	3.5×10^{-3}	5.2×10^{-3}	3.6×10^{-4}	1.1×10^{-4}
5	3.5×10^{-2}	7.9×10^{-2}	7.3×10^{-3}	7.9×10^{-6}	2.8×10^{-2}	9.8×10^{-6}
6	2.4×10^{-2}	5.4×10^{-2}	3.1×10^{-4}	1.4×10^{-5}	4.8×10^{-2}	4.9×10^{-5}
7	1.4×10^{-3}	4.2×10^{-2}	6.5×10^{-3}	1.8×10^{-4}	1.0×10^{-3}	3.7×10^{-5}
8	3.9×10^{-2}	2.2×10^{-4}	9.4×10^{-4}	6.9×10^{-4}	2.5×10^{-4}	3.4×10^{-4}
9	5.6×10^{-1}	1.4×10^{-2}	1.8×10^{-3}	4.5×10^{-3}	2.4×10^{-4}	2.8×10^{-5}
10	3.4×10^{-2}	4.2×10^{-2}	9.9×10^{-4}	3.5×10^{-4}	2.8×10^{-4}	4.1×10^{-5}

Table 7: Closed loop MSE on *out-dis-test* data from ten different training runs for each model for the diffusion-sorption equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	4.7×10^{-2}	2.3×10^{-2}	2.9×10^{-3}	-	1.3×10^{-2}	9.7×10^{-5}
2	2.1×10^{-1}	4.3×10^{-2}	1.6×10^{-3}	-	1.1×10^{-2}	1.5×10^{-5}
3	9.2×10^{-2}	5.5×10^{-2}	3.4×10^{-3}	-	1.4×10^{-2}	9.5×10^{-5}
4	2.0×10^{-1}	1.9×10^{-2}	2.0×10^{-3}	-	1.1×10^{-2}	3.4×10^{-5}
5	4.4×10^{-2}	1.3×10^{-1}	4.6×10^{-4}	-	2.1×10^{-2}	4.9×10^{-6}
6	3.9×10^{-4}	1.0×10^{-1}	8.7×10^{-3}	-	4.4×10^{-2}	1.9×10^{-5}
7	9.3×10^{-4}	9.1×10^{-2}	1.6×10^{-2}	-	1.5×10^{-2}	1.5×10^{-5}
8	5.0×10^{-3}	2.6×10^{-3}	9.5×10^{-4}	-	1.2×10^{-2}	1.0×10^{-4}
9	4.6×10^{-1}	3.4×10^{-2}	7.1×10^{-3}	-	1.3×10^{-2}	1.2×10^{-5}
10	4.1×10^{-2}	8.0×10^{-2}	2.6×10^{-3}	-	1.3×10^{-2}	1.6×10^{-5}

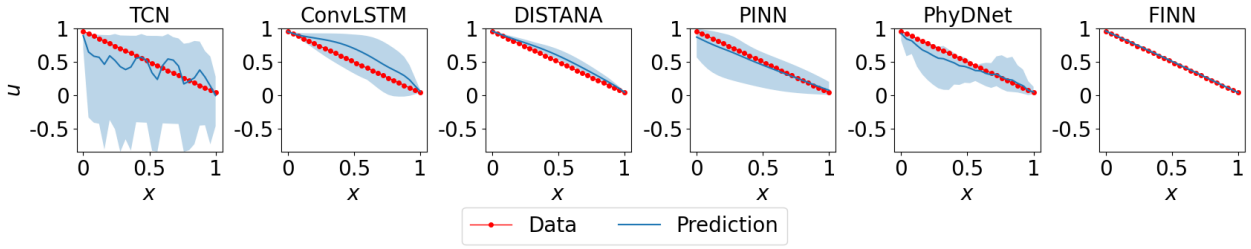


Figure 9: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration in the diffusion-sorption equation at $t = 10\,000$ for the *in-dis-test* dataset.

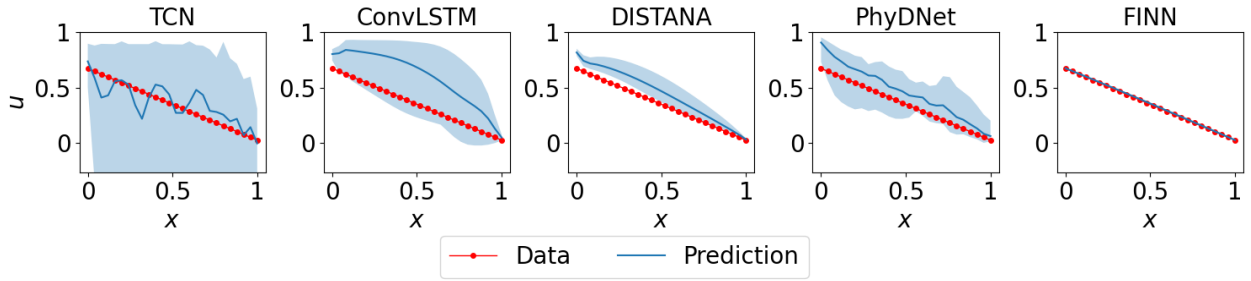


Figure 10: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration in the diffusion-sorption equation at $t = 10\,000$ for the *out-dis-test* dataset.

Data The 1D diffusion-sorption equation is written as the following coupled system of equations:

$$\frac{\partial u}{\partial t} = \frac{D}{R(u)} \frac{\partial^2 u}{\partial x^2}, \quad (16) \qquad \frac{\partial u_t}{\partial t} = D\phi \frac{\partial^2 u}{\partial x^2}, \quad (17)$$

where the dissolved concentration u and total concentration u_t are the main unknowns. The effective diffusion coefficient is denoted by $D = 5 \times 10^{-4}$, the retardation factor is $R(u)$, a function of u, and the porosity is denoted by $\phi = 0.29$.

In this work, the Freundlich sorption isotherm, see details in (Nowak & Guthke, 2016), was chosen to define the retardation factor:

$$R(u) = 1 + \frac{1-\phi}{\phi} \rho_s k n_f u^{n_f - 1}, \quad (18)$$

where $\rho_s = 2\,880$ is the bulk density, $k = 3.5 \times 10^{-4}$ is the Freundlich's parameter, and $n_f = 0.874$ is the Freundlich's exponent.

The simulation domain for the ***train* data** is defined with $x = [0, 1]$, $t = [0, 2\,500]$ and is discretized with $N_x = 26$ spatial locations, and $N_t = 501$ simulation steps. The initial condition is defined as $u(x, 0) = 0$, and the boundary condition is defined as $u(0, t) = 1.0$ and $u(1, t) = D\frac{\partial u}{\partial x}$.

The ***in-dis-test* data** was simulated with $x = [0, 1]$ and with the time span of $t = [2\,500, 10\,000]$ and $N_t = 2\,001$. Initial condition is taken from the train data at $t = 2\,500$ and boundary conditions are also identical to the train data.

The simulation domain for the ***out-dis-test* data** was identical with the *in-dis-test* data, except for the boundary condition that was defined as $u(0, t) = 0.7$.

Model Architectures **TCN** is designed to have two input neurons, four hidden layers of size [4, 8, 8, 8], and two output neurons. **ConvLSTM** has two input- and output neurons and one hidden layer with 16 neurons. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and two, respectively, while a hidden layer of size 12 is used. The pure ML models were trained on the first 400 time steps and validated on the remaining 100 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done during teacher forcing. **PINN** was defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 2]. **PhyDNet** was defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer. For **FINN**, the modules $\varphi_{\mathcal{N}}$ and $\varphi_{\mathcal{D}}$ were used, with $\varphi_{\mathcal{D}}$ defined as a feedforward network with the size of [1, 10, 20, 10, 1]

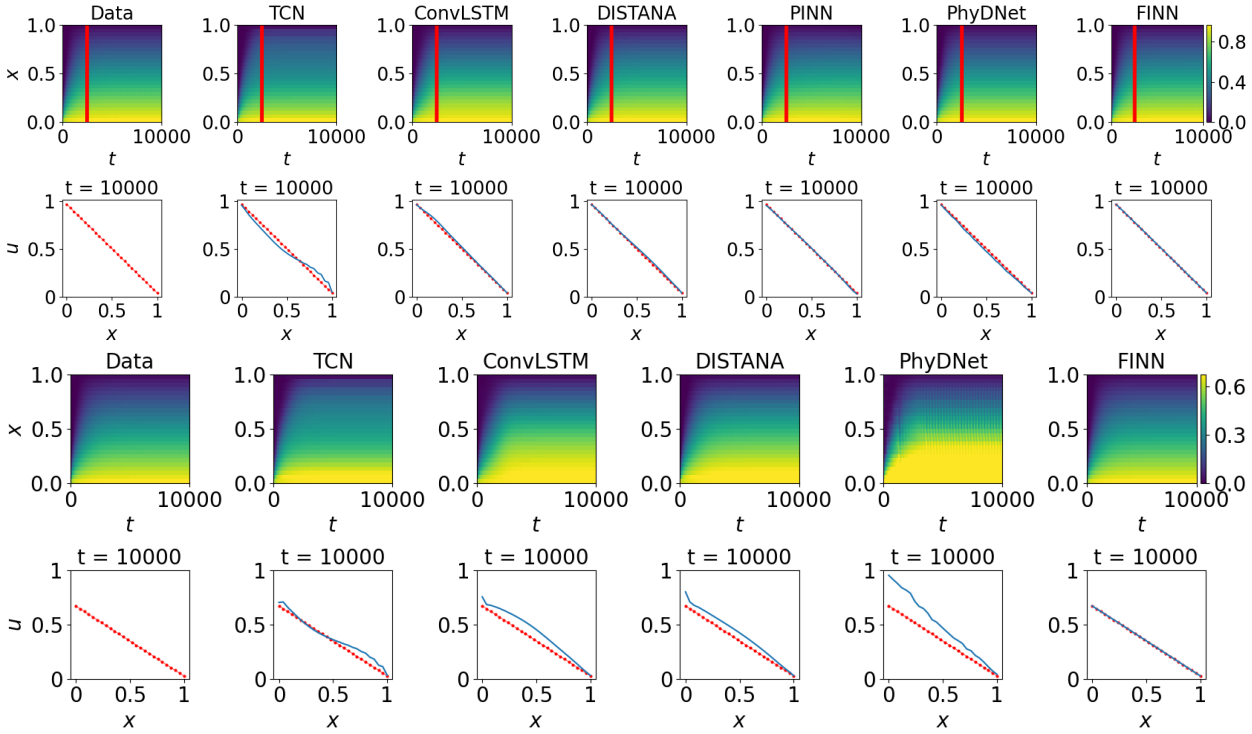


Figure 11: Plots of diffusion-sorption's dissolved concentration (red) and prediction (blue) using different models. The first and third rows show the solution over x and t (vertical red lines mark the transition from *train* to *in-dis-test*). Second (*in-dis-test*) and fourth (*out-dis-test*) rows visualize the solution distributed in x at $t = 10\,000$.

that takes u as an input and outputs the retardation factor $R(u)$. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 5), *in-dis-test* (Table 6 and Figure 9), and *out-dis-test* (Table 7 and Figure 10) datasets. Results for the total concentration u_t were omitted due to high similarity to the concentration of the reported contamination solution u.

C.3. Diffusion-Reaction

The diffusion-reaction equation is applicable in physical and biological systems, for example in pattern formation (Turing, 1952).

Data In the current paper, we consider the 2D diffusion-reaction for that class of problems:

$$\frac{\partial u_1}{\partial t} = R_1(u_1, u_2) + D_1 \left(\frac{\partial^2 u_1}{\partial x^2} + \frac{\partial^2 u_1}{\partial y^2} \right), \tag{19}$$

$$\frac{\partial u_2}{\partial t} = R_2(u_1, u_2) + D_2 \left(\frac{\partial^2 u_2}{\partial x^2} + \frac{\partial^2 u_2}{\partial y^2} \right). \tag{20}$$

Here, $D_1 = 10^{-3}$ and $D_2 = 5 \times 10^{-3}$ are the diffusion coefficient for the activator and inhibitor, respectively. The system of equations is coupled through the reaction terms $R_1(u_1, u_2)$ and $R_2(u_1, u_2)$ which are both dependent on u_1 and u_2. In this work, the Fitzhugh-Nagumo (Klaasen & Troy, 1984) system was considered to define the reaction function:

$$R_1(u_1, u_2) = u_1 - u_1^3 - k - u_2, \tag{21}$$

$$R_2(u_1, u_2) = u_1 - u_2, \tag{22}$$

with $k = 5 \times 10^{-3}$.

The simulation domain for the ***train*** **data** is defined with $x = [-1, 1]$, $y = [-1, 1]$, $t = [0, 10]$ and is discretized with $N_x = 49$ and $N_y = 49$ spatial locations, and $N_t = 101$ simulation steps. The initial condition was defined as $u_1(x, 0) = u_2(x, 0) = \sin(\pi(x + 1)/2) \sin(\pi(y + 1)/2)$,

Table 8: Closed loop MSE on *train* data from ten different training runs for each model for the diffusion-reaction equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	8.7×10^{-3}	1.7×10^{-3}	1.7×10^{-3}	2.7×10^{-4}	8.5×10^{-5}	1.0×10^{-4}
2	4.4×10^{-3}	1.6×10^{-3}	4.2×10^{-3}	5.6×10^{-4}	7.1×10^{-5}	1.6×10^{-4}
3	3.2×10^{-2}	1.4×10^{-3}	9.1×10^{-3}	2.1×10^{-4}	5.8×10^{-5}	1.4×10^{-4}
4	8.7×10^{-3}	7.8×10^{-4}	6.5×10^{-4}	2.8×10^{-4}	6.8×10^{-5}	1.2×10^{-4}
5	2.5×10^{-2}	1.5×10^{-3}	4.7×10^{-4}	2.8×10^{-4}	7.6×10^{-5}	1.0×10^{-4}
6	8.0×10^{-3}	4.8×10^{-4}	2.1×10^{-4}	2.5×10^{-4}	6.5×10^{-5}	1.7×10^{-4}
7	2.4×10^{-2}	3.7×10^{-3}	8.6×10^{-3}	6.1×10^{-5}	8.2×10^{-5}	1.6×10^{-4}
8	1.1×10^{-2}	1.0×10^{-3}	2.4×10^{-3}	1.0×10^{-4}	7.8×10^{-5}	1.0×10^{-4}
9	1.3×10^{-2}	2.2×10^{-3}	3.8×10^{-3}	3.2×10^{-5}	8.8×10^{-5}	1.5×10^{-4}
10	5.6×10^{-3}	7.3×10^{-2}	8.6×10^{-3}	6.2×10^{-4}	7.4×10^{-5}	1.1×10^{-4}

Table 9: Closed loop MSE on *in-dis-test* data from ten different training runs for each model for the diffusion-reaction equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.8×10^{-1}	8.0×10^{-2}	1.6×10^{-1}	7.7×10^{-2}	6.2×10^{-2}	2.9×10^{-3}
2	5.9×10^{-1}	5.7×10^{-2}	1.9×10^{-1}	2.3×10^{-1}	8.8×10^{-2}	1.6×10^{-3}
3	6.5×10^{-1}	9.2×10^{-2}	3.0×10^{-1}	3.5×10^{-2}	7.8×10^{-2}	1.7×10^{-3}
4	3.1×10^{-1}	6.7×10^{-2}	2.0×10^{-1}	2.6×10^{-2}	6.8×10^{-2}	2.9×10^{-3}
5	7.3×10^{-1}	4.9×10^{-2}	1.9×10^{-1}	1.3×10^{-1}	5.3×10^{-2}	1.7×10^{-3}
6	1.9×10^{-1}	5.4×10^{-2}	2.8×10^{-2}	8.9×10^{-2}	9.9×10^{-2}	2.0×10^{-3}
7	6.0×10^{-1}	1.0×10^{-1}	2.2×10^{-1}	5.4×10^{-2}	6.7×10^{-2}	1.5×10^{-3}
8	7.1×10^{-1}	8.6×10^{-2}	1.9×10^{-1}	1.3×10^{-2}	9.1×10^{-2}	2.3×10^{-3}
9	5.5×10^{-1}	2.3×10^{-1}	1.6×10^{-1}	2.3×10^{-2}	6.7×10^{-2}	1.8×10^{-3}
10	2.2×10^{-1}	1.2×10^{-1}	1.7×10^{-1}	2.6×10^{-2}	1.1×10^{-1}	2.1×10^{-3}

Table 10: Closed loop MSE on *out-dis-test* data from ten different training runs for each model for the diffusion-reaction equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	2.3×10^{-2}	3.2×10^{-3}	9.4×10^{-3}	-	3.9×10^{-2}	6.5×10^{-3}
2	1.7×10^{-1}	1.3×10^{-2}	1.3×10^{-2}	-	5.4×10^{-2}	5.5×10^{-3}
3	2.5×10^{-1}	1.3×10^{-2}	2.1×10^{-2}	-	2.5×10^{-2}	5.9×10^{-3}
4	7.3×10^{-2}	7.2×10^{-3}	7.9×10^{-3}	-	4.2×10^{-2}	6.5×10^{-3}
5	2.6×10^{-1}	5.9×10^{-3}	2.6×10^{-3}	-	2.8×10^{-2}	6.1×10^{-3}
6	6.4×10^{-2}	1.5×10^{-2}	2.1×10^{-3}	-	2.9×10^{-2}	6.0×10^{-3}
7	2.2×10^{-1}	1.0×10^{-2}	1.9×10^{-2}	-	2.8×10^{-2}	5.7×10^{-3}
8	2.0×10^{-1}	3.9×10^{-3}	1.2×10^{-2}	-	2.1×10^{-2}	6.1×10^{-3}
9	2.3×10^{-1}	4.3×10^{-2}	1.3×10^{-2}	-	2.1×10^{-2}	6.0×10^{-3}
10	3.6×10^{-2}	3.7×10^{-2}	3.3×10^{-2}	-	5.9×10^{-2}	6.2×10^{-3}

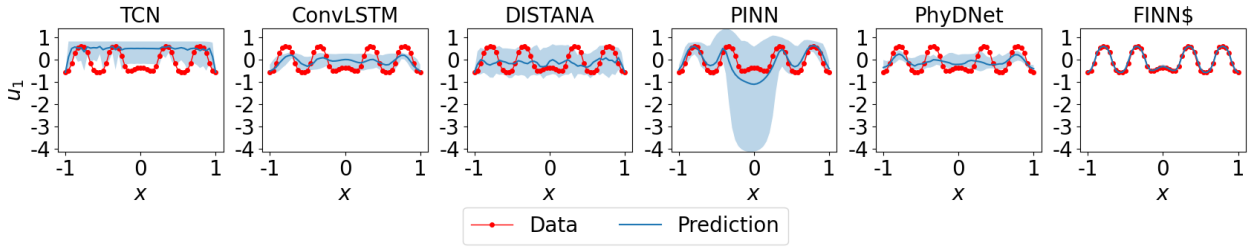


Figure 12: Prediction mean over ten different trained models (with 95% confidence interval) of the activator in the diffusion-reaction equation at $t = 50$ for the *in-dis-test* dataset.

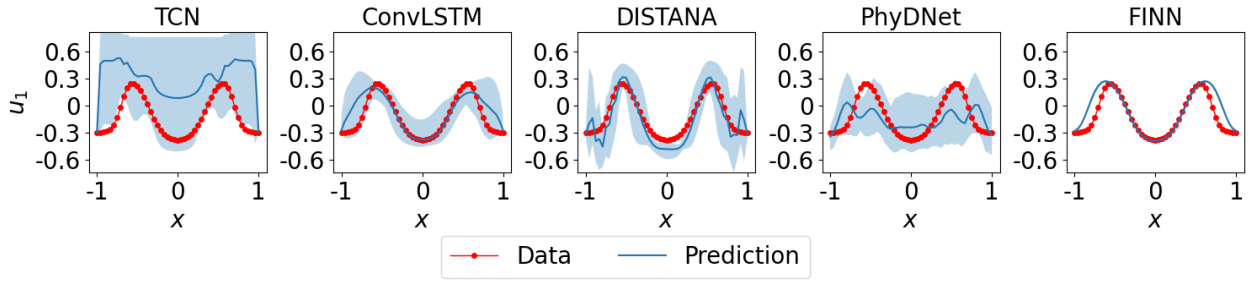


Figure 13: Prediction mean over ten different trained models (with 95% confidence interval) of the activator in the diffusion-reaction equation at $t = 10$ for the *out-dis-test* dataset.

and the corresponding boundary condition was defined as $\frac{\partial u_1}{\partial x}(-1, t) = 0$, $\frac{\partial u_1}{\partial x}(1, t) = 0$, $\frac{\partial u_2}{\partial x}(-1, t) = 0$, and $\frac{\partial u_2}{\partial x}(1, t) = 0$.

The ***in-dis-test*** **data** is simulated with with $x = [-1, 1]$, $y = [-1, 1]$ and with the time span of $t = [10, 50]$ and $N_t = 501$. Initial condition is taken from the train data at $t = 10$ and boundary conditions are also identical to the train data.

The simulation domain for the ***out-dis-test*** **data** was identical with the train data, except for the initial condition that was defined as $u_1(x, 0) = u_2(x, 0) = \sin(\pi(x + 1)/2) \sin(\pi(y + 1)/2) - 0.5$, i.e. subtracting 0.5 from the original initial condition.

Model Architectures **TCN** is designed to have two input- and output neurons, and one hidden layer of size 32. **ConvLSTM** has two input- and output neurons and one hidden layer of size 24. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and two, respectively, while a hidden layer of size 32 is used. The pure ML models were trained on the first 70 time steps and validated on the remaining 30 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done during teacher forcing. **PINN** is defined as a feedforward network with the size of [3, 20, 20, 20, 20, 20, 20, 20, 20, 2]. **PhyDNet** is defined with the PhyCell containing 32 input dimensions, 49 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer. For **FINN**, the modules $\varphi_{\mathcal{N}}$, $\varphi_{\mathcal{D}}$ and Φ_ψ are used, with $\varphi_{\mathcal{D}}$ set as two learnable scalars that learn the diffusion coefficients D_1 and D_2, and Φ_ψ defined as a feedforward network with the size of [2, 20, 20, 20, 2] that takes u_1 and u_2 as inputs and outputs the reaction functions $R_1(u_1, u_2)$ and $R_2(u_1, u_2)$. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 8), *in-dis-test* (Table 9 and Figure 12), and *out-dis-test* (Table 10 and Figure 13) datasets. Results for the total inhibitor u_2 were omitted due to high similarity to the reported activator u_1.

C.4. Allen-Cahn

The Allen-Cahn equation is commonly employed in reaction-diffusion systems, in particular to model phase separation in multi-component alloy systems (Raissi et al., 2019).

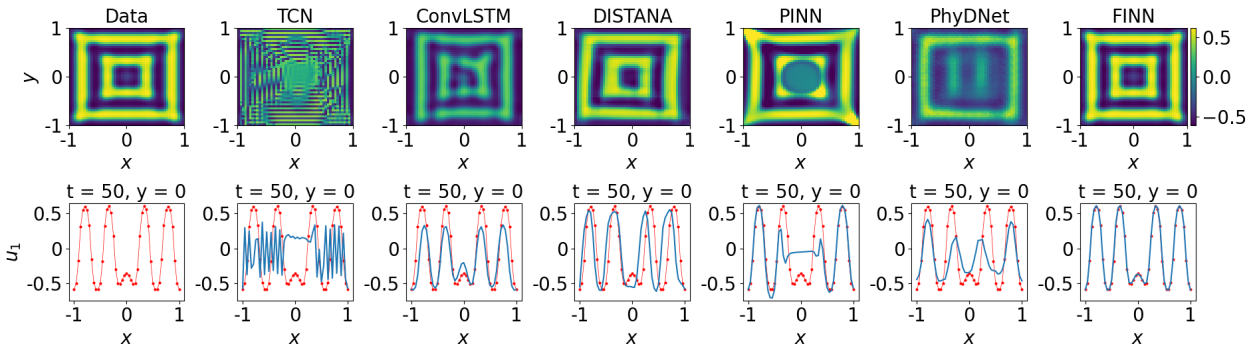


Figure 14: Plots of diffusion-reaction's activator data u_1 (red) and extrapolated prediction (blue) using different models. The plots in the first row show the solution distributed over x and y at $t = 50$, and the plots in the second row show the solution distributed in x at $y = 0$ and $t = 50$.

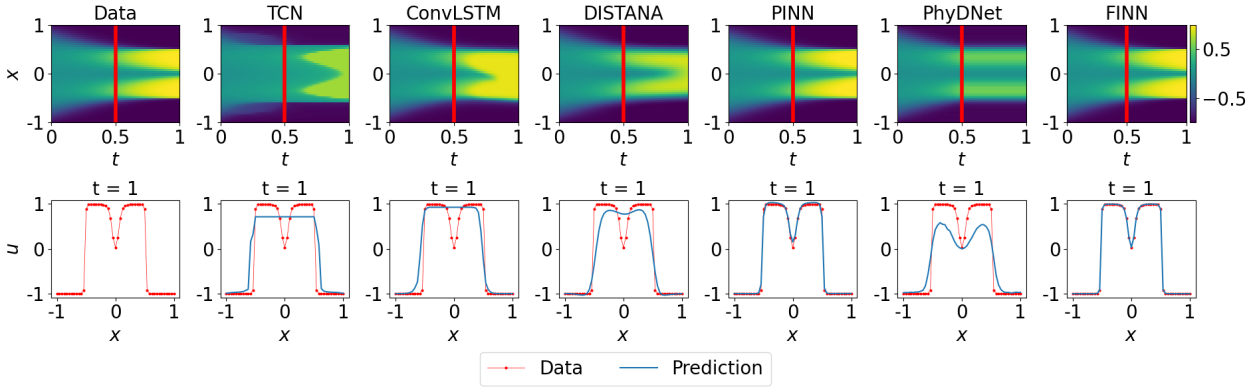


Figure 15: Plots of Allen-Cahn's data and prediction of *in-dis-test* data using different models. The plots in the first row show the solution over x and t (the red lines mark the transition from *train* to *in-dis-test*), the second row visualizes the best model's solution distributed in x at $t = 1$.

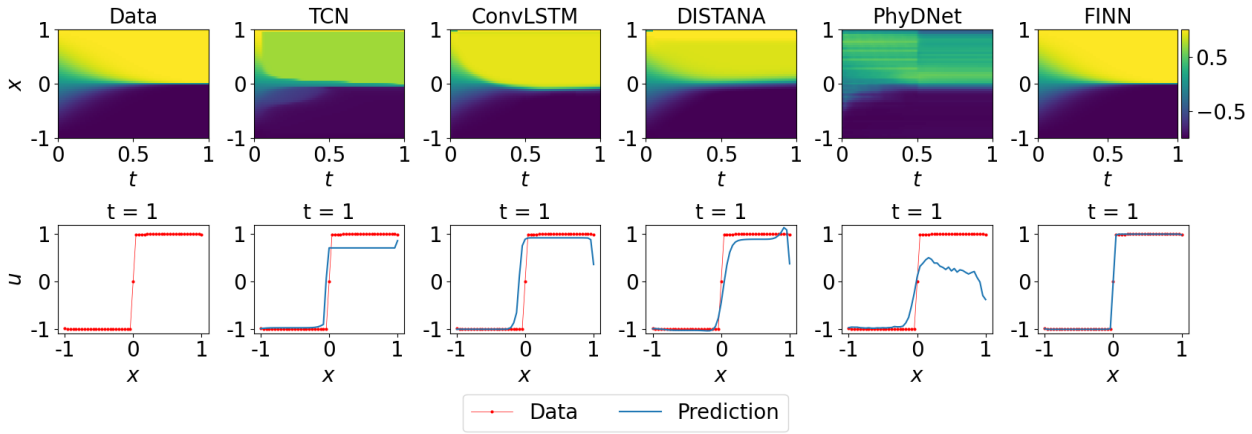


Figure 16: Plots of Allen-Cahn's data and prediction of *out-dis-test* data using different models. The plots in the first row show the solution over x and t, the second row visualizes the solution distributed in x at $t = 1$.

Table 11: Closed loop MSE on the *train* data from ten different training runs for each model for the Allen-Cahn equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.1×10^{-2}	7.1×10^{-2}	2.9×10^{-4}	1.6×10^{-5}	1.5×10^{-4}	1.8×10^{-5}
2	4.5×10^{-2}	3.0×10^{-2}	4.7×10^{-4}	4.2×10^{-5}	6.4×10^{-5}	8.2×10^{-6}
3	2.9×10^{-2}	1.2×10^{-2}	8.0×10^{-4}	4.2×10^{-5}	2.5×10^{-4}	5.4×10^{-6}
4	8.9×10^{-2}	6.5×10^{-3}	4.9×10^{-4}	1.6×10^{-6}	4.8×10^{-5}	4.7×10^{-7}
5	1.4×10^{-2}	8.0×10^{-2}	2.6×10^{-4}	1.3×10^{-5}	5.1×10^{-5}	2.3×10^{-5}
6	1.1×10^{-2}	4.6×10^{-3}	5.7×10^{-4}	6.4×10^{-6}	4.7×10^{-5}	2.5×10^{-6}
7	9.6×10^{-3}	7.1×10^{-3}	4.2×10^{-4}	1.4×10^{-5}	1.7×10^{-4}	1.1×10^{-5}
8	3.6×10^{-2}	5.3×10^{-4}	8.5×10^{-4}	2.1×10^{-5}	3.7×10^{-5}	2.9×10^{-7}
9	6.9×10^{-1}	4.0×10^{-1}	7.6×10^{-4}	2.0×10^{-5}	9.8×10^{-5}	3.0×10^{-7}
10	2.8×10^{-2}	1.8×10^{-4}	1.5×10^{-4}	3.8×10^{-5}	9.8×10^{-5}	2.9×10^{-7}

Table 12: Closed loop MSE on *in-dis-test* data from ten different training runs for each model for the Allen-Cahn equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	9.9×10^{-2}	1.2×10^{0}	4.8×10^{-2}	7.0×10^{-3}	1.4×10^{-1}	5.6×10^{-5}
2	2.9×10^{-1}	8.6×10^{-1}	9.1×10^{-2}	1.8×10^{-1}	8.2×10^{-2}	2.4×10^{-5}
3	2.3×10^{-1}	3.6×10^{-1}	7.9×10^{-2}	4.3×10^{-1}	1.0×10^{-1}	1.6×10^{-5}
4	4.7×10^{-1}	3.4×10^{-1}	7.4×10^{-2}	4.6×10^{-3}	9.9×10^{-2}	2.1×10^{-6}
5	1.9×10^{-1}	5.5×10^{-1}	7.3×10^{-2}	2.2×10^{-3}	8.5×10^{-2}	9.3×10^{-5}
6	1.2×10^{-1}	3.2×10^{-1}	1.0×10^{-1}	2.0×10^{-3}	1.2×10^{-1}	1.1×10^{-5}
7	1.2×10^{-1}	5.4×10^{-1}	1.3×10^{-1}	5.0×10^{-4}	2.3×10^{-1}	3.4×10^{-5}
8	3.0×10^{-1}	5.6×10^{-1}	9.5×10^{-2}	1.3×10^{-1}	1.0×10^{-1}	1.3×10^{-6}
9	1.4×10^{0}	1.3×10^{0}	1.6×10^{-1}	1.2×10^{-2}	1.0×10^{-1}	1.3×10^{-6}
10	2.3×10^{-1}	5.5×10^{-2}	4.6×10^{-2}	3.3×10^{-2}	9.4×10^{-2}	1.3×10^{-6}

Table 13: Closed loop MSE on *out-dis-test* data from ten different training runs for each model for the Allen-Cahn equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	4.1×10^{-2}	5.3×10^{-2}	1.4×10^{-2}	-	6.2×10^{-1}	7.4×10^{-5}
2	2.0×10^{-1}	3.8×10^{-1}	3.7×10^{-2}	-	2.4×10^{-1}	2.9×10^{-5}
3	1.3×10^{-1}	1.5×10^{-1}	6.2×10^{-2}	-	6.2×10^{-1}	1.8×10^{-5}
4	2.0×10^{-1}	2.9×10^{-1}	1.1×10^{-2}	-	7.3×10^{-1}	3.0×10^{-6}
5	1.7×10^{-1}	8.3×10^{-2}	4.0×10^{-2}	-	6.8×10^{-1}	1.2×10^{-4}
6	8.7×10^{-2}	2.9×10^{-1}	3.1×10^{-2}	-	7.8×10^{-1}	1.5×10^{-5}
7	1.0×10^{-1}	1.5×10^{-1}	5.9×10^{-2}	-	1.0×10^{0}	4.3×10^{-5}
8	1.5×10^{-1}	5.3×10^{-1}	9.1×10^{-2}	-	5.4×10^{-1}	1.9×10^{-6}
9	1.2×10^{0}	1.4×10^{0}	1.0×10^{-1}	-	6.2×10^{-1}	1.8×10^{-6}
10	9.6×10^{-2}	7.5×10^{-2}	1.5×10^{-2}	-	5.9×10^{-1}	1.9×10^{-6}

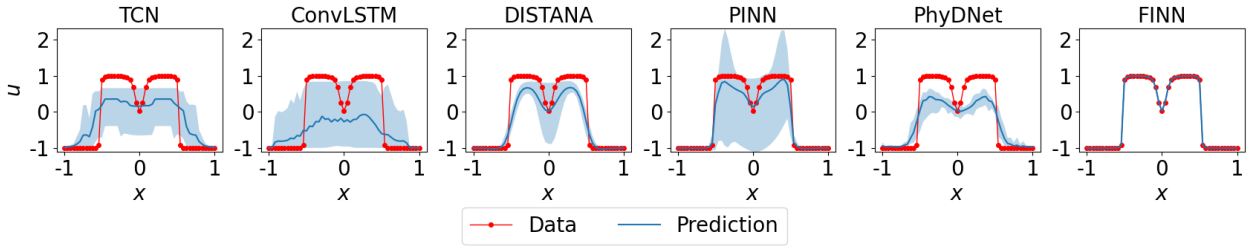


Figure 17: Prediction mean over ten different trained models (with 95% confidence interval) of the Allen-Cahn equation at $t = 1$ for the *in-dis-test* dataset.

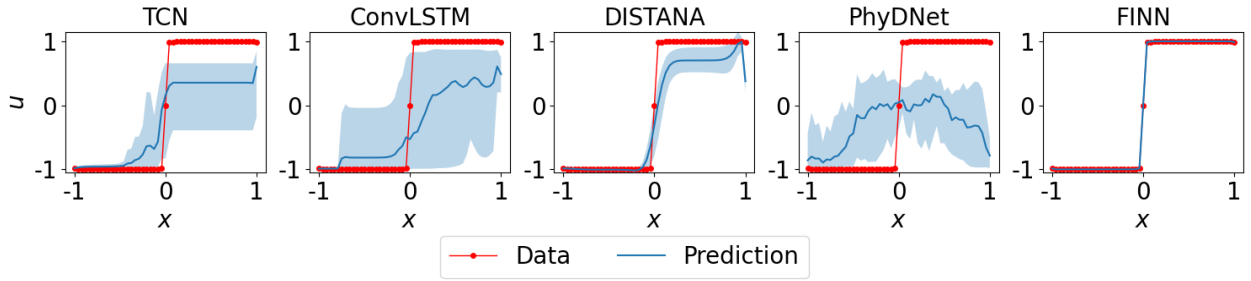


Figure 18: Prediction mean over ten different trained models (with 95% confidence interval) of the Allen-Cahn equation at $t = 1$ for the *out-dis-test* data.

Data The 1D Allen-Cahn equation is written as

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + R(u), \tag{23}$$

where the main unknown is u, the reaction term is denoted as $R(u)$ which is a function of u and the diffusion coefficient is $D = 10^{-4}$. In the current work, the reaction term is defined as:

$$R(u) = 5u - 5u^3, \tag{24}$$

to reproduce the experiment conducted in the PINN paper (Raissi et al., 2019).

The simulation domain for the ***train*** **data** is defined with $x = [-1, 1]$, $t = [0, 0.5]$ and is discretized with $N_x = 49$ spatial locations, and $N_t = 201$ simulation steps. The initial condition is defined as $u(x, 0) = x^2 \cos(\pi x)$, and periodic boundary condition is used, i.e. $u(-1, t) = u(1, t)$.

In-dis-test **data** is simulated with $x = [-1, 1]$ and a time span of $t = [0.5, 1]$ and $N_t = 401$. Initial condition is taken from the train data at $t = 0.5$ and boundary conditions are also similar to the train data.

The simulation domain for the ***out-dis-test*** **data** is identical with the train data, except for the initial condition that is defined as $u(x, 0) = \sin(\pi x/2)$.

Model Architectures Both **TCN** and **ConvLSTM** are designed to have one input neuron, one hidden layer of size 32, and one output neuron. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and a hidden layer of size 32 is used. The pure ML models were trained on the first 150 time steps and validated on the remaining 50 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done during teacher forcing. **PINN** was defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 1] (8 hidden layers, each contains 20 hidden neurons). **PhyDNet** was defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer.

For **FINN**, the modules $\varphi_{\mathcal{N}}$, $\varphi_{\mathcal{D}}$, and Φ_{ψ} were used, with $\varphi_{\mathcal{D}}$ defined as a learnable scalar that learns the diffusion coefficient D, and Φ_{ψ} defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes u as an input and outputs the reaction function $R(u)$. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 11), *in-dis-test* (Table 12, Figure 15

and Figure 17), and *out-dis-test* (Table 13, Figure 16 and Figure 18) datasets.

C.5. Soil Parameters and Simulation Domains for the Experimental Dataset

Identical to Praditia et al. (2021), the soil parameters and simulation (and experimental) domain used in the real-world diffusion-sorption experiment are summarized in Table 14 for core samples #1, #2, and #2B.

For all experiments, the core samples are subjected to a constant contaminant concentration at the top u_s, which can be treated as a Dirichlet boundary condition numerically. Notice that, for core sample #2, we set u_s to be slightly higher to compensate for the fact that there might be fractures at the top of core sample #2, so that the contaminant can break through the core sample faster.

For core samples #1 and #2, Q is the flow rate of clean water at the bottom reservoir that determines the Cauchy boundary condition at the bottom of the core samples. For core sample #2B, note that the sample length is significantly longer than the other samples, and by the end of the experiment, no contaminant has broken through the core sample. Therefore, we assume the bottom boundary condition to be a no-flow Neumann boundary condition; see (Praditia et al., 2021) for details.

Table 14: Soil and experimental parameters of core samples #1, #2, and #2B. D is the diffusion coefficient, ϕ is the porosity, ρ_s is the bulk density, L and r are the length and radius of the sample, t_{end} is the simulation time, Q is the flow rate in the bottom reservoir and u_s is the total concentration of trichloroethylene in the sample.

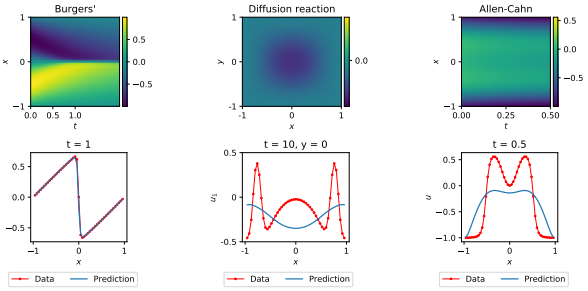
Soil parameters				
Parameter	Unit	Core #1	Core #2	Core #2B
D	m^2/day	2.00×10^{-5}	2.00×10^{-5}	2.78×10^{-5}
ϕ	-	0.288	0.288	0.288
ρ_s	kg/m^3	1957	1957	1957
Simulation domain				
Parameter	Unit	Core #1	Core #2	Core #2B
L	m	0.0254	0.02604	0.105
r	m	0.02375	0.02375	N/A
t_{end}	days	38.81	39.82	48.88
Q	m^3/day	1.01×10^{-4}	1.04×10^{-4}	N/A
u_s	kg/m^3	1.4	1.6	1.4

D. Ablations

D.1. Baseline Assessment with Polynomial Regression

In this section, we apply polynomial regression to show that the example problems chosen in this work (i.e. Burgers', diffusion-sorption, diffusion-reaction, and Allen-Cahn) are not easy to solve. First, we use polynomial regression to fit the unknown variable $u = f(x, t)$, similar to PINN. Figure 20 shows the prediction of u for each example, obtained using the fitted polynomial coefficients. For the Burgers' equation, the *train* and *in-dis-test* predictions have MSE values of 5.0×10^{-2} and 4.1×10^{-2}, respectively. For the diffusion-sorption equation, the *train* and *in-dis-test* predictions have MSE values of 3.0×10^{-3} and 4.1×10^{1}, respectively. For the diffusion-reaction equation, the *train* and *in-dis-test* predictions have MSE values of 1.7×10^{-2} and 3.9×10^{4}, respectively. For the Allen-Cahn equation, the *train* and *in-dis-test* predictions have MSE values of 8.8×10^{-3} and 2.2×10^{3}, respectively. The simple polynomial fitting fails to obtain accurate predictions of the solution for all example problems. The results also show that the polynomials overfit the data, evidenced by the significant deterioration of performance during extrapolation (prediction of *in-dis-test* data). The diffusion-reaction and the Allen-Cahn equations are particularly the most difficult to fit, because they require higher order polynomials to obtain reasonable accuracy. With the high order, they still fail to even fit the *train* data well.

Next, we also consider using polynomial fitting in lieu of ANNs (namely the modules $\varphi_{\mathcal{A}}$, $\varphi_{\mathcal{D}}$, and Φ_{ψ}) in FINN. Indeed, with this method, the model successfully predicts Burgers' equation (Figure 19a). The MSE values are 1.9×10^{-4}, 1.0×10^{-3}, and 1.3×10^{-4} for *train*, *in-dis-test*, and



(a) (b) (c)

Figure 19: Plots of the *train* prediction in the Burgers' (left), diffusion-reaction (center), and Allen-Cahn equations (right) using FINN with polynomial fitting. Due to instability issues, the diffusion-sorption equation could not be solved with the polynomial FINN. The plots in the first row show the solution over x and t, and the plots in the second row show the solution distributed in x.

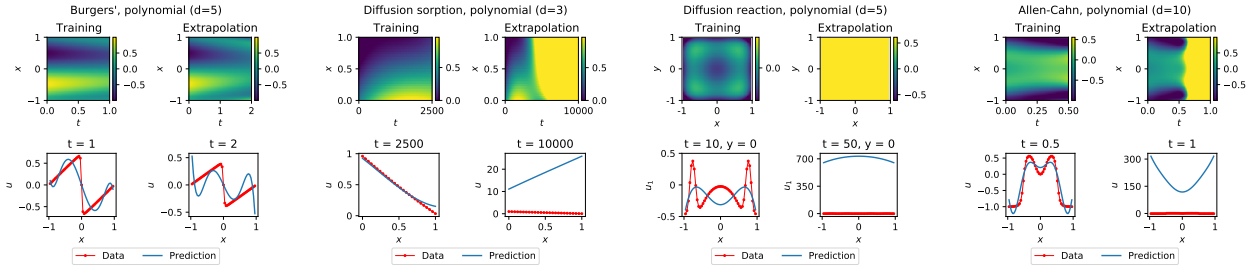
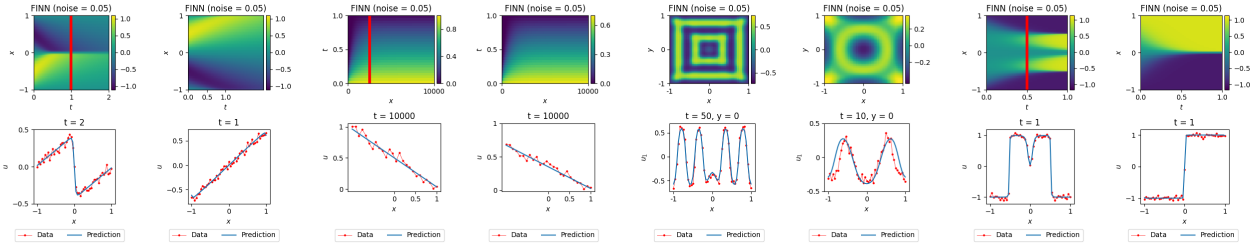


Figure 20: Prediction pairs for *train* and *in-dis-test* (left and right columns of the pairs) of the Burgers' (left, polynomial order 5), diffusion-sorption (second left, polynomial order 3), diffusion-reaction (second right, polynomial order 5 since higher orders did not converge), and Allen-Cahn (right, polynomial order 10) equations using polynomial regression.



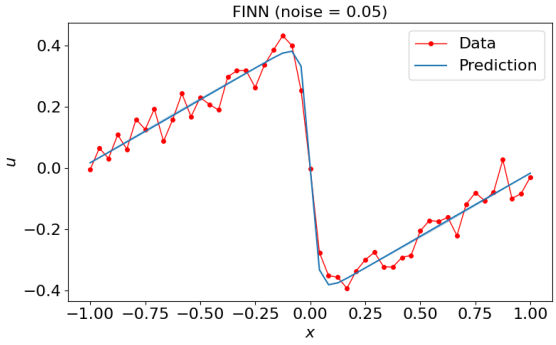
(a) Burgers' (b) Diffusion-sorption (c) Diffusion-reaction (d) Allen-Cahn

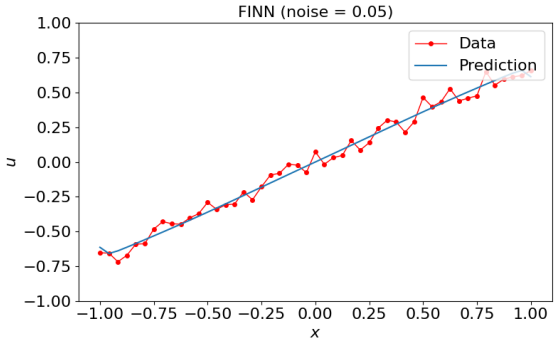
Figure 21: Paired plots of the *in-dis-test* and *out-dis-test* prediction (left and right of the pairs, respectively) after training FINN with noisy data. The plots in the first row of the pairs show the solution over x and t (red line marks the transition from *train* to *in-dis-test*), and the plots in the second row of the pairs show the best model's solution distributed in x.

out-dis-test data, respectively. However, the model fails to complete the training for the diffusion-sorption equation due to major instabilities (the polynomials can produce negative output and therefore, negative diffusion coefficient, leading to numerical instability). Moreover, the model also fails to sufficiently learn the diffusion-reaction (Figure 19b) and the Allen-Cahn (Figure 19c) equations. For the diffusion-reaction equation, the MSE values are 2.5×10^{-2}, 1.7×10^{-1}, and 4.8×10^{-2} for *train*, *in-dis-test*, and *out-dis-test* data, respectively. For the Allen-Cahn equation, the MSE values are 5.6×10^{-2}, 2.5×10^{-1}, and 4.3×10^{-1} for *train*, *in-dis-test*, and *out-dis-test* data, respectively. Even though the unknown equations do not seem too complicated, they are still difficult to solve together with the PDE as a whole. The results show that for these particular problems, ANNs serve better because they allow better control during training in form of constraints, and they produce more regularized outputs than high order polynomials. However, while ANNs are not unique in their selection, they are most convenient for our implementation.

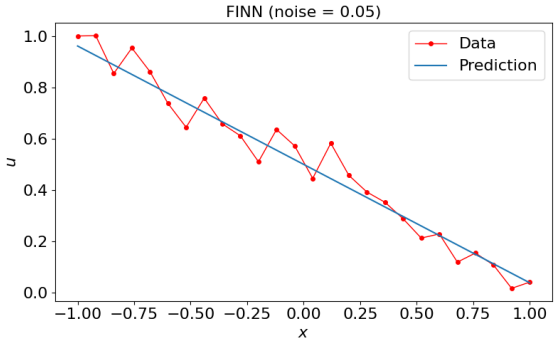
D.2. Noise Robustness Test of FINN

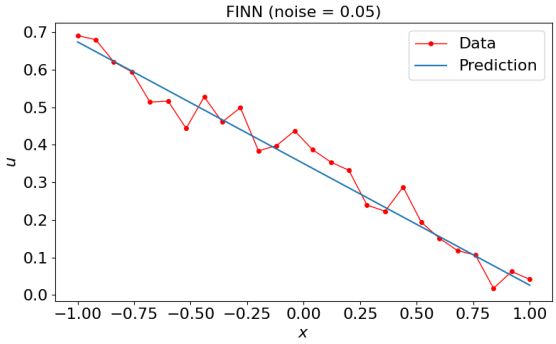
In this section, we test the robustness of FINN when trained using noisy data. All the synthetic data is generated with the same parameters, only added with noise with the distribution $\mathcal{N}(0.0, 0.05)$. For the Burgers' equation (Figure 21a and Figure 22a), the average MSE values are $2.5 \times 10^{-3} \pm 4.1 \times 10^{-6}$, $2.4 \times 10^{-3} \pm 6.6 \times 10^{-6}$, and $2.5 \times 10^{-3} \pm 4.0 \times 10^{-6}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. For the diffusion-sorption equation (Figure 21b and Figure 22b), the average MSE values are $2.5 \times 10^{-3} \pm 4.5 \times 10^{-6}$, $2.5 \times 10^{-3} \pm 3.7 \times 10^{-6}$, and $2.5 \times 10^{-3} \pm 3.7 \times 10^{-6}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. For the diffusion-reaction equation (Figure 21c and Figure 22c), the average MSE values are $3.2 \times 10^{-3} \pm 5.3 \times 10^{-4}$, $1.6 \times 10^{-2} \pm 8.8 \times 10^{-3}$, and $8.3 \times 10^{-3} \pm 4.9 \times 10^{-4}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. For the Allen-Cahn equation (Figure 21d and Figure 22d), the average MSE values are $2.5 \times 10^{-3} \pm 1.1 \times 10^{-6}$, $2.5 \times 10^{-3} \pm 9.1 \times 10^{-6}$, and $2.5 \times 10^{-3} \pm 6.3 \times 10^{-6}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. These results show that even though FINN is trained with noisy data, it is still able to capture the essence of the equation and generalize well to different initial and boundary conditions. Additionally, the prediction is consistent, shown by the low values of the MSE standard deviation, as well as the very narrow confidence interval in the plots.



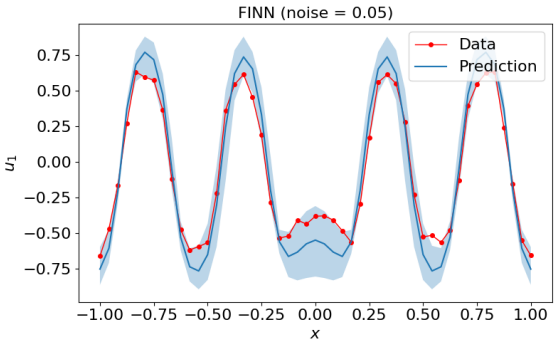


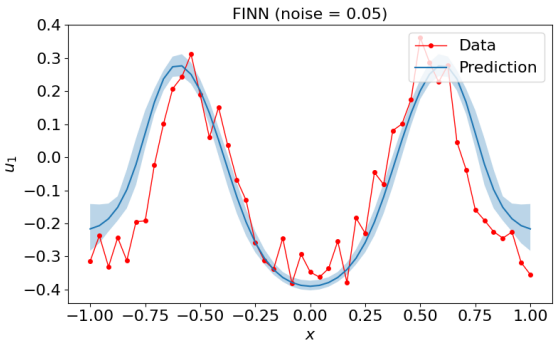
(a) Burgers'



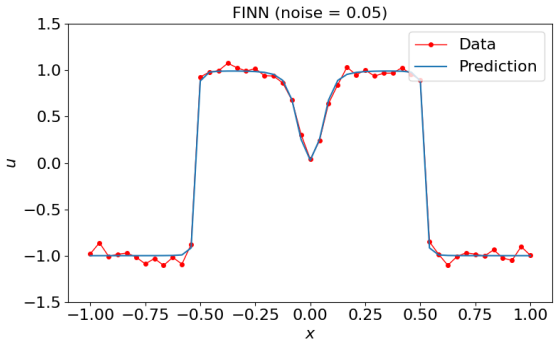


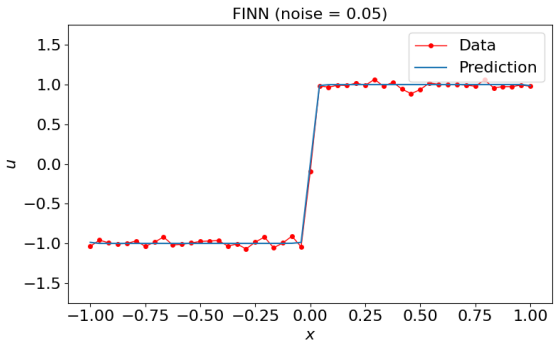
(b) Diffusion-sorption





(c) Diffusion-reaction





(d) Allen-Cahn

Figure 22: Prediction mean over ten different trained FINN (with 95% confidence interval) of the Burgers' (top left), diffusion-sorption (top right), diffusion-reaction (bottom left), and Allen-Cahn (bottom right) equations obtained by training FINN with noisy data for the *in-dis-test* and *out-dis-test* (top and bottom, accordingly) prediction.

D.3. Neural ODE Ablation and Runtime Analysis

In order to stress the role of Neural ODE in FINN, we conduct an ablation by removing the Neural ODE part and directly feeding the output of FINN as input in the next time. This amounts to traditional closed loop training or an Euler ODE integration scheme, where the model directly predicts and outputs the delta from $u^{(t)}$ to $u^{(t+1)}$ (also referred to as residual training). Results are summarized in Table 15 and unveil two major effects of Neural ODE.

First, it helps stabilizing the training reasonably: We observe FINN having immense difficulties without Neural ODE at learning the diffusion-sorption equation (not even completing a single epoch without producing `NaN` values). Similar but less dramatic failures were observed on the Burgers', diffusion-reaction, and Allen-Cahn equations, where FINN without Neural ODE on average only managed 50, 72, and 85 epochs, respectively, without producing `NaN` values. This consolidates the supportive aspect of the Neural ODE component, which we explain is due to its adaptive time-stepping feature. More importantly, the adaptive time-stepping of Neural ODE makes it possible to apply FINN to experimental data in the first place, where the time deltas between successive measurements are not constant.

Second and as to be expected, the runtime when adding Neural ODE in the computations increases significantly. We compare the runtime for each model, run on a CPU with i9-9900K core, a clock speed of 3.60 GHz, and 32 GB RAM. Additionally, we also perform the comparison of GPU runtime on a GTX 1060 (i.e. with 6 GB VRAM). The results are summarized in Table 16. Note that the purpose of this comparison is not an optimized benchmark, but only to show that the runtime of FINN is comparable with the other models, especially when run in CPU. When run on GPU, however, FINN runs slightly slower. This is caused by the fact that FINN's implementation is not yet optimized for GPU. More importantly, the Neural ODE package we use benefits only from a larger batch size. As shown in Figure 23, GPU is faster for batch size larger than 20 000, whereas the maximum size that we use in the example is 2 401. With smaller batch size, CPU usage is faster.

In general, only the PINN model has a benefit when computed on the GPU, since the function is only called once on all batches. This is different for all other models that have to unroll a prediction of the sequence into the future recurrently (except for TCN which is a convolution approach that is faster on GPU). Accordingly, The overhead of copying the tensors to GPU outweighs the GPU's parallelism benefit, compared to directly processing the sequence on the CPU iteratively. On the two-dimensional diffusion-reaction benchmark, the GPU's speed-up comes into play, since in here, the simulation domain is discretized into $49 \times 49 = 2401$ volumes (compared to 49 for the Burgers' and Allen-Cahn, and 26 for the diffusion-sorption equations). Note that we have observed significantly varying runtimes on different GPUs (i.e. up to three seconds for FINN on Burgers' on an RTX 3090), which might be caused by lack of support of certain packages for a particular hardware, but further investigation is required.

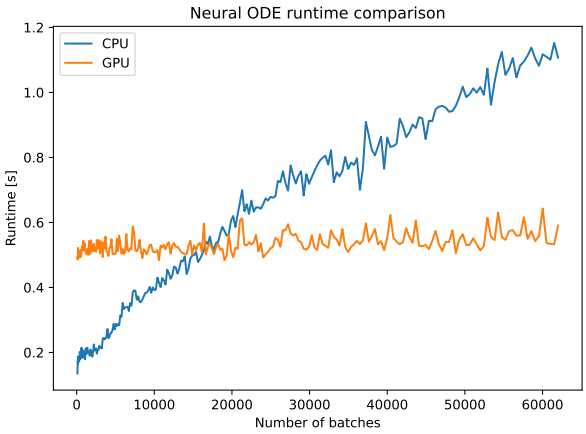


Figure 23: Runtime comparison of Neural ODE with a single hidden layer consisting of 50 hidden nodes run on CPU and GPU for 1 000 time steps. The benefit of using GPU starts to be seen when larger batch sizes ($> 20\,000$) are used.

Additionally, we want to emphasize that the higher runtime of FINN on GPU is not caused by the time step adaptivity. In fact, employing the adaptive time stepping strategy is cheaper than choosing all time steps to be small enough (to guarantee numerical stability). As we learn a PDE, we have no exact knowledge to derive a dedicated time integration scheme, but the adaptive Runge-Kutta method is one of the best generic choices. As our PDE and its characteristics change during training, time step adaptivity is a real asset, because for example the Courant–Friedrichs–Lewy (CFL) condition (Courant et al., 1967) would consistently change throughout the training. Therefore, the time step adaptivity is not a bottleneck, but rather a solution for a more efficient computation.

Furthermore, in relation to FINN's limitation (see subsection B.2) topics like numerical stabilization schemes for larger time steps, adaptive spatial grid, optimized implementation in an High Performance Computing (HPC) setting, parallelization, etc. are still very interesting for future works.

Table 15: Comparison of MSE on the *Train* data for individual runs with and without Neural ODE (NODE) on the different equations. Average of models without (w/o) NODE is calculated over all individual runs. Average scores of models with (w/) NODE are taken from Table 1. Left and right columns for the different equations show the MSE and epoch from which on `NaN` values occured, respectively. FINN was trained for 100 epochs, overall, i.e. 100 in the second columns corresponds to a training without any `NaN` values encountered.

Run	Burgers'		Diffusion-sorption		Diffusion-reaction		Allen-Cahn	
1	6.0×10^{-6}	100	-	0	2.5×10^{-2}	2	6.6×10^{-6}	100
2	7.9×10^{-6}	100	-	0	1.3×10^{-3}	14	6.1×10^{-5}	13
3	9.9×10^{-4}	100	-	0	6.3×10^{-4}	27	1.1×10^{-5}	100
4	1.4×10^{-5}	100	-	0	3.9×10^{-3}	78	4.7×10^{-7}	100
5	1.3×10^{-3}	2	-	0	4.3×10^{-3}	100	8.4×10^{-5}	100
6	5.9×10^{-6}	100	-	0	1.8×10^{-2}	83	2.6×10^{-7}	100
7	4.6×10^{-6}	100	-	0	8.2×10^{-4}	31	1.6×10^{-7}	100
8	1.2×10^{-3}	3	-	0	1.0×10^{-3}	15	5.5×10^{-7}	100
9	1.4×10^{-5}	13	-	0	2.3×10^{-4}	100	5.5×10^{-7}	100
10	5.1×10^{-6}	100	-	0	1.8×10^{-4}	45	7.3×10^{-6}	39
Avg w/o NODE	$(3.5 \pm 5.2) \times 10^{-4}$	72	-	0	$(5.5 \pm 8.3) \times 10^{-3}$	50	$(1.1 \pm 1.8) \times 10^{-5}$	85
Avg w/ NODE	$(2.8 \pm 2.9) \times 10^{-6}$	100	(omitted)	100	$(1.3 \pm 0.3) \times 10^{-4}$	100	$(6.9 \pm 7.7) \times 10^{-6}$	100

Table 16: Comparison of runtime (in seconds) of single forward passes between different deep learning (above dashed line) and physics-aware neural network (below dashed line) methods on the different equations.

Eqn.	Model	CPU (i9-9900K, 3.60 GHz)	GPU (GTX 1060)
Burger (1D)	TCN	0.423	0.130
	ConvLSTM	0.052	0.079
	DISTANA	0.059	0.098
	PINN	0.036	0.007
	PhyDNet	0.107	0.192
	FINN	0.066	0.161
Diffusion-sorption (1D)	TCN	1.228	0.393
	ConvLSTM	0.119	0.194
	DISTANA	0.145	0.223
	PINN	0.073	0.010
	PhyDNet	0.263	0.475
	FINN	0.676	1.638
Diffusion-reaction (2D)	TCN	14.15	0.800
	ConvLSTM	0.230	0.052
	DISTANA	0.190	0.051
	PINN	1.647	0.787
	PhyDNet	0.159	0.113
	FINN	0.342	0.330
Allen-Cahn (1D)	TCN	0.442	0.128
	ConvLSTM	0.050	0.082
	DISTANA	0.060	0.097
	PINN	0.035	0.007
	PhyDNet	0.108	0.191
	FINN	0.028	0.071

Table 17: MSE of PINN trained using data with finer resolution from ten different training runs for the diffusion-reaction equation.

Run	*Train*	*In-dis-test*	*Out-dis-test*
1	1.4×10^{-4}	7.8×10^{-2}	-
2	1.8×10^{-4}	5.6×10^{-2}	-
3	8.6×10^{-5}	1.5×10^{-2}	-
4	1.0×10^{-3}	1.5×10^{-1}	-
5	5.2×10^{-5}	2.7×10^{-2}	-
6	2.9×10^{-4}	8.1×10^{-1}	-
7	1.3×10^{-4}	3.4×10^{-2}	-
8	8.9×10^{-5}	2.9×10^{-2}	-
9	3.9×10^{-5}	1.3×10^{-2}	-
10	3.3×10^{-5}	1.7×10^{-2}	-

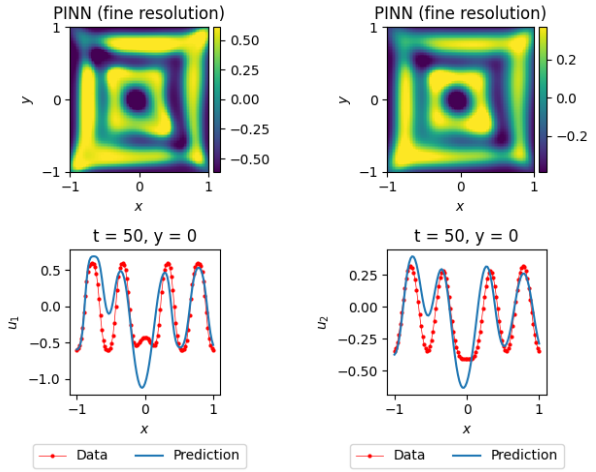


Figure 24: Plots of the diffusion-reaction equation's activator u using PINN trained with finer resolution dataset. *In-dis-test* prediction (left) and *out-dis-test* (right)

D.4. Training PINN With Finer Spatial Resolution

In order to determine whether the reduced accuracy of PINN in our experiments was caused by a coarse spatial resolution (we only used 49, 26 and 49×49 spatial positions at Burger, diffusion-sorption, and diffusion-reaction, respectively), another experiment was conducted per target equation where the spatial resolution was increased to $N_x = 999$, $N_x = 251$, and $N_x = 99$, $N_y = 99$, respectively. As reported in Table 17, Figure 24, and Figure 25, the performance increased slightly but by far did not reach FINN's accuracy. Identical results were achieved in the Burgers' and diffusion-sorption equations but are omitted due to high conceptual similarity.

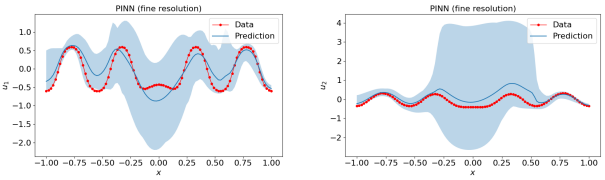


Figure 25: PINN fine resolution. Prediction mean (with 95% confidence interval) of the activator (left) and inhibitor (right) in the diffusion-reaction equation at $t = 50$ compared with the *in-dis-test* data.

Table 18: MSE of PhyDNet using the original network size from ten different training runs for the diffusion-reaction equation.

Run	*Train*	*In-dis-test*	*Out-dis-test*
1	9.3×10^{-5}	1.8×10^{-1}	7.6×10^{-2}
2	2.9×10^{-5}	6.5×10^{-2}	3.5×10^{-2}
3	2.2×10^{-5}	6.7×10^{-2}	4.0×10^{-2}
4	4.6×10^{-5}	7.2×10^{-2}	3.9×10^{-2}
5	3.5×10^{-5}	6.5×10^{-2}	8.8×10^{-2}
6	5.7×10^{-5}	1.2×10^{-1}	1.0×10^{-1}
7	3.5×10^{-5}	5.7×10^{-2}	9.0×10^{-2}
8	3.5×10^{-5}	5.9×10^{-2}	2.6×10^{-2}
9	2.3×10^{-5}	6.3×10^{-2}	4.1×10^{-2}
10	3.5×10^{-5}	6.1×10^{-2}	8.1×10^{-2}

D.5. PhyDNet With Original Amount of Parameters

To verify whether our reduction of parameters and the removal of the encoder and decoder layers caused PhyDNet to perform worse, we repeated the experiments for the three equations of interest using the original PhyDNet architecture as proposed in (Guen & Thome, 2020). However, our results indicate no significant changes in performance, as reported in Table 18, Figure 26, and Figure 27. Again, results for the inhibitor u_2 as well as for the Burgers' and diffusion-reaction equations were omitted due to high conceptual similarity.

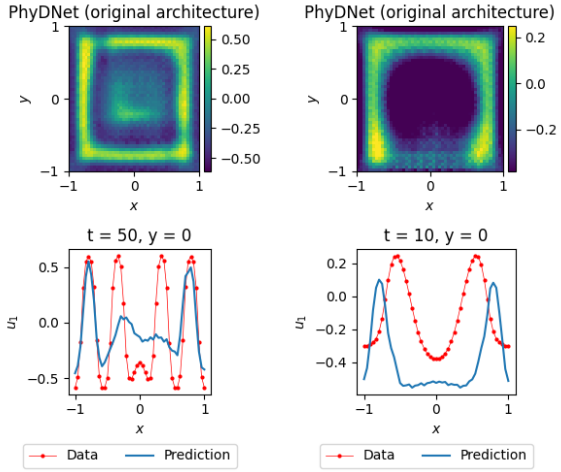


Figure 26: Plots of the diffusion-reaction equation's activator u using PhyDNet with the original network size. *In-dis-test* prediction (left) and *out-dis-test* (right).

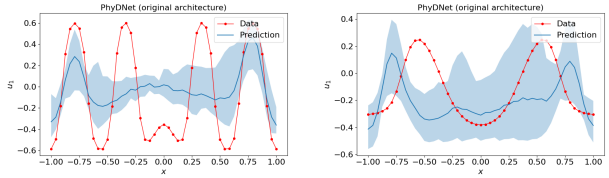


Figure 27: PhyDNet original architecture. Prediction mean (with 95% confidence interval) of the activator (left) and inhibitor (right) in the diffusion-reaction equation at $t = 50$ compared with the *in-dis-test* data.

D Publication 4: Learning Groundwater Contaminant Diffusion-Sorption Processes with a Finite Volume Neural Network

AGU ADVANCING EARTH AND SPACE SCIENCE

Water Resources Research

RESEARCH ARTICLE
10.1029/2022WR033149

Learning Groundwater Contaminant Diffusion-Sorption Processes With a Finite Volume Neural Network

Timothy Praditia[1], Matthias Karlbauer[2], Sebastian Otte[2], Sergey Oladyshkin[1], Martin V. Butz[2], and Wolfgang Nowak[1]

[1]Department of Stochastic Simulation and Safety Research for Hydrosystems, Institute for Modelling Hydraulic and Environmental Systems, University of Stuttgart, Stuttgart, Germany, [2]Neuro-Cognitive Modeling Group, Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Tübingen, Germany

Key Points:

- We propose a hybrid modeling framework combining uncertainty-quantifying artificial neural networks with physical domain knowledge
- Our method learns diffusion processes and discovers unknown sorption isotherms, while incorporating different boundary conditions
- Excellent generalization is retained even when trained with sparse and noisy real-world data from a laboratory experiment

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:

T. Praditia,
timothy.praditia@iws.uni-stuttgart.de

Citation:

Praditia, T., Karlbauer, M., Otte, S., Oladyshkin, S., Butz, M. V., & Nowak, W. (2022). Learning groundwater contaminant diffusion-sorption processes with a finite volume neural network. *Water Resources Research*, *58*, e2022WR033149. https://doi.org/10.1029/2022WR033149

Received 14 JUL 2022
Accepted 13 NOV 2022

Author Contributions:

Conceptualization: Timothy Praditia, Matthias Karlbauer, Sebastian Otte, Sergey Oladyshkin, Martin V. Butz, Wolfgang Nowak
Data curation: Timothy Praditia, Matthias Karlbauer
Formal analysis: Timothy Praditia, Matthias Karlbauer, Sebastian Otte, Sergey Oladyshkin, Martin V. Butz, Wolfgang Nowak
Funding acquisition: Martin V. Butz, Wolfgang Nowak

Abstract Improved understanding of complex hydrosystem processes is key to advance water resources research. Nevertheless, the conventional way of modeling these processes suffers from a high conceptual uncertainty, due to almost ubiquitous simplifying assumptions used in model parameterizations/closures. Machine learning (ML) models are considered as a potential alternative, but their generalization abilities remain limited. For example, they normally fail to predict accurately across different boundary conditions. Moreover, as a black box, they do not add to our process understanding or to discover improved parameterizations/closures. To tackle this issue, we propose the hybrid modeling framework FINN (finite volume neural network). It merges existing numerical methods for partial differential equations (PDEs) with the learning abilities of artificial neural networks (ANNs). FINN is applied on discrete control volumes and learns components of the investigated system equations, such as numerical stencils, model parameters, and arbitrary closure/constitutive relations. Consequently, FINN yields highly interpretable results. We demonstrate FINN's potential on a diffusion-sorption problem in clay. Results on numerically generated data show that FINN outperforms other ML models when tested under modified boundary conditions, and that it can successfully differentiate between the usual, known sorption isotherms. Moreover, we also equip FINN with uncertainty quantification methods to lay open the total uncertainty of scientific learning, and then apply it to a laboratory experiment. The results show that FINN performs better than calibrated PDE-based models as it is able to flexibly learn and model sorption isotherms without being restricted to choose among available parametric models.

1. Introduction

Scientists and engineers have been trying to model physical phenomena occurring in nature for centuries, one of which is the transport of a quantity in space and time through natural media. A few examples include: subsurface fluid flow modeling (e.g., Ghosh et al., 2020; T. Koch et al., 2021), climate modeling (e.g., IPCC, 2013; Marchuk, 1974), and diffusion-reaction modeling (e.g., Turing, 1952; Wei & Winter, 2017). Of course, contaminant transport and attenuation in water resources research also falls into this problem class. Problems of this type are usually described mathematically using partial differential equations (PDEs) because of their space and time dependencies.

However, despite promising development in computing power and availability of data, the behavior of several of these physical systems is still poorly understood (Winsberg, 2003). As such, simplifying assumptions are required to model parts of the processes that are either still unknown, too complicated, or act on scales much smaller than those of interest. A concrete example is the simplification of diffusion-sorption problems to be modeled with sorption isotherms (Al-Ghouti & Da'ana, 2020; Limousin et al., 2007). The available sorption isotherms are valid only under certain geochemical conditions. In particular, they are not applicable to more complicated diffusion-sorption problems that may, for example, involve reactive transport. Other examples include the choice of relative permeability and saturation relationships in multiphase flow in porous media (K. Li & Horne, 2006; Moghadasi et al., 2015), and the reaction formulations in diffusion-reaction systems (Allen & Cahn, 1979; Klaasen & Troy, 1984).

To address this issue, we propose the finite volume neural network (FINN) as a novel physics-aware machine learning (ML) modeling framework. FINN combines the well-established numerical discretization strategy of the finite volume method (FVM) and the flexibility and learning ability of artificial neural networks (ANNs). Most importantly, this combination allows to explicitly and accurately learn parts of the unknown or poorly understood

Investigation: Timothy Praditia, Matthias Karlbauer
Methodology: Timothy Praditia, Matthias Karlbauer, Sebastian Otte, Sergey Oladyshkin, Martin V. Butz, Wolfgang Nowak
Project Administration: Martin V. Butz, Wolfgang Nowak
Resources: Timothy Praditia, Matthias Karlbauer
Software: Timothy Praditia, Matthias Karlbauer
Supervision: Sebastian Otte, Sergey Oladyshkin, Martin V. Butz, Wolfgang Nowak
Validation: Timothy Praditia, Matthias Karlbauer
Visualization: Timothy Praditia, Matthias Karlbauer
Writing – original draft: Timothy Praditia, Matthias Karlbauer
Writing – review & editing: Sebastian Otte, Sergey Oladyshkin, Martin V. Butz, Wolfgang Nowak

processes mentioned above, while maintaining numerical stability, producing highly accurate predictions, and providing scientifically interpretable functions of interest. For the sake of demonstration, we focus here on diffusion and sorption of groundwater contaminant in fully saturated clay. This process is relevant, for example, in clay liners of landfills (Hendry et al., 2003; Timms et al., 2018) or in long-term tailing of groundwater pollution (Huang & Goltz, 2015; Johnson et al., 2003).

One particularly harmful contaminant is trichloroethylene (TCE), which is categorized as a carcinogen (National Toxicology Program, US Department of Health and Human Services, 2021); yet it is still commonly used in industry (World Health Organization, Regional Office for World Health Organization Regional Office for Europe, 2000). TCE is a dense non-aqueous phase liquid (Pankow & Cherry, 1996), meaning that it is denser than water and has a very low solubility in water. As a consequence, when TCE infiltrates into the subsurface, it migrates downwards until it reaches an impermeable barrier, where it forms a pool, resulting in difficult remediation (e.g., G. H. Brown et al., 2012; J. Koch & Nowak, 2015). One of the most common impermeable barriers in the subsurface is a layer of clay. However, even though a layer of clay is impermeable, TCE can still diffuse into it and over time can contaminate the groundwater in the vicinity (e.g., Nowak & Guthke, 2016; Pankow & Cherry, 1996). It is accordingly necessary to build a model of such processes in order to predict the longevity of contamination, select remediation strategies, and assess environmental and health risks.

The process of TCE adsorption on clay surfaces is influenced by a complex mechanism (Allen-King et al., 1996; Pankow & Cherry, 1996). Consequently, simplifying assumptions have to be made that introduce conceptual uncertainties in the modeling process, such as the choice of particular isotherms (Limousin et al., 2007), the unknown parameters of those sorption isotherms (e.g., Nowak & Guthke, 2016), uncertain clay/soil parameters, effective diffusion coefficients of dissolved chemicals in water (Hayduk & Laudie, 1974; Wilke & Chang, 1955), as well as uncertain initial and boundary conditions that the model requires to be satisfied. As a consequence of these uncertainties, we are faced with a model choice problem (Höge et al., 2019). Furthermore, all available models are inherently generated with simplifying assumptions to different extents, enhancing the model choice problem even more. Thus, a more flexible way of modeling is needed, such that the unknown true process is covered by the chosen modelling strategy. Accordingly, we propose a data-driven modeling approach, which induces physics-aware inductive biases to effectively learn the unknown actual process properties.

In order to approach the question of conceptual model learning, it is worth looking at the rapidly evolving field of ML, which has revolutionized various domains, including image and language processing (Krizhevsky et al., 2012; T. B. Brown et al., 2020). Recently, ML is also being applied to approximate physical processes, such as rigid body interactions, liquid propagation, or weather and sea-surface temperature prediction (Battaglia et al., 2016; De Bézenac et al., 2019; Espeholt et al., 2021; Lienen & Günnemann, 2022; Rasp et al., 2020; Sanchez-Gonzalez et al., 2020). The benefit and charm of applying ML models lies in their ability to learn an input-to-output mapping function without any knowledge of the underlying process that describes the data (i.e., the "true model"). Furthermore, ML models also have the potential to learn complicated functional relationships that are not addressed in the physical models due to limited computational power or lack of understanding of the modeled systems. In the following, we summarize the related works, separating them in non-physics-motivated (*pure ML*), *physics-motivated*, and *physics-aware ML* models.

An exemplary *pure ML* method for processing spatiotemporal data is the temporal convolution network (TCN) proposed by Lea et al. (2016), which performs convolution operations along space and time dimensions. TCNs have been proven successful in various classification tasks (Bai et al., 2018; Kalchbrenner et al., 2016). On the other hand, their applicability as autoregressive models in generative forecasting tasks is limited (Almqvist, 2019; Karlbauer et al., 2020). In contrast, ConvLSTM (Shi et al., 2015) is a spatiotemporal recurrent neural network that processes temporal data points sequentially. It is therefore slower than TCN's parallel convolution operations. It can, however, aggregate and conserve any past information in a latent state, implementing a flexible, latent memory structure. In contrast, the temporal horizon of TCNs is limited to the receptive field, that is, the number of time steps fed into the TCN filters. Moreover, because ConvLSTMs are by training optimized towards maintaining stable predictions within a recurrent loop, they typically exhibit superior performance on related, for example, autoregressive tasks (Almqvist, 2019; Karlbauer et al., 2020). However, the freedom of pure ML models has several limitations when learning an unconstrained function that should reflect a physical process. First, they typically depend on large amounts of data in order to learn a useful mapping. Second, they can be expected to

behave adequately only within the range of the data they have been trained on. And third, they can produce physically implausible predictions.

These limitations can be addressed by incorporating structural physical knowledge to formulate *physics-motivated ML* models, using the form of (relational) inductive biases (Battaglia et al., 2018). DISTANA (Karlbauer et al., 2019), for example, shares similarities with ConvLSTM, albeit with advanced and physically motivated lateral information flow between neighboring control volumes. An alternative approach is to learn (Z. Li et al., 2020a, 2020b) operators using neural networks, both in the input and output space (DeepONet, L. Lu et al., 2021), as well as in the frequency domain (Fourier neural operator model, FNO, Z. Li et al., 2020a). By design, these methods are implemented to specifically learn PDEs from data, but are not guided or constrained by physical principles that would be already known.

Recently, much effort was directed to finding reasonable ways of connecting the learning abilities of ANNs not only with structural, but also with functional knowledge, resulting in *physics-aware ML* methods. The physics-informed neural network (PINN), for example, explicitly learns to solve a *given* equation, such as Burgers' or Navier-Stokes, in order to accelerate simulation (Jin et al., 2021; Raissi et al., 2019). PINN has also been applied to solve subsurface fluid flow problems (Tartakovsky et al., 2020) and to perform data assimilation for parameter estimation, accounting for multiple physical processes (Q. He et al., 2020). The physics-informed prior from PINN has also recently been embedded in FNO models, resulting in a Physics-Informed Neural Operator (PINO) model (Z. Li et al., 2021). Other methods do not depend on receiving the underlying equation, but approximate it implicitly through data. These weaker physics constraints are either implemented by means of convolution-like operators representing derivatives up to a particular degree, for example, PDE-Net (Long et al., 2018), PhyDNet (consisting of a data-driven ConvLSTM and a physics-constrained path) proposed by Guen and Thome (2020), or SIREN (Sitzmann et al., 2020); or by directly learning the transition function $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ (e.g., in form of a vector field) that maps the d-dimensional observation in frame t to the succeeding frame $t + 1$ (De Bézenac et al., 2019; Tran & Ward, 2017). More recently, graph-based approaches are formulated by Seo et al. (2019) and Salehi and Giannacopoulos (2021) to explicitly consider differences between neighboring control volumes on spatially irregularly distributed data.

Nevertheless, a common downside of all these approaches is the missing facility to include explicit physical knowledge—such as the *structure* of a particular PDE—into the learning process. In contrast, and similar to our work, Bar-Sinai et al. (2019); Kochkov et al. (2021); and Zhuang et al. (2021) propose learning selected parts of ordinary differential equations (ODEs), but focus more on accelerating supersampling procedures and less on predictive, explorative, and explainability tasks. APHYNITY (Yin et al., 2020) represents an alternative approach, where traditional physical models are augmented by ML methods, effectively learning to minimize the residual between an explicitly stated physical model and the observation. A survey of methods that combine physics with ML has been proposed by Karniadakis et al. (2021) and an extensive collection is maintained by Thuerey et al. (2021).

Despite these exciting developments in the areas of physics-motivated and physics-aware ML modeling, important issues remain to be addressed. That is, building PINN models requires the complete knowledge of the modeled systems, including the aforementioned closure/constitutive relationships, which are usually the main source of uncertainty in the modeling process. As a consequence, PINN can be trained on incorrect equations, if spurious assumptions are chosen. Additionally, most, if not all, of the spatiotemporal ML models adopt convolutional operations to process the spatial correlation between data points. Convolutional operations, however, can only pad constant values on the domain boundaries. Therefore, such ML models have no means to implement sufficient, non-constant boundary conditions. For example, they are not able to properly incorporate boundary conditions that depend on derivatives such as the Neumann or Cauchy boundary conditions. Furthermore, the existing ML models struggle when confronted with different initial or boundary conditions. In such a case, typically retraining is necessary, which requires large amounts of observation data that is often expensive and difficult to obtain. In short, the existing physics-motivated and physics-aware ML models are either too restricted by the physical knowledge or too lenient so that they learn relationships that do not exist (i.e., they overfit).

Another crucial drawback of most ML models is the lack of practical uncertainty quantification (UQ) applications, despite the availability of numerous theoretical foundations (Jospin et al., 2022). This is mainly caused by computational challenges of existing UQ algorithms for large ML models. When dealing with real-world applications, however, UQ is critical. For example, when performing a risk assessment about a poorly understood

system based on uncertain models trained on uncertain (noisy) and sparse data (Nowak & Guthke, 2016; Wöhling et al., 2015; Xu et al., 2020). Moreover, fundamental scientific work depends on hypothesis-testable models, which is strongly supported by models with quantifiable uncertainty. Consequently, it is important to design a model with few parameters and an interpretable structure to enable feasible implementations of available UQ algorithms.

The goal of this work, therefore, is to provide a framework that merges physics-aware ML models with well-selected structures known from numerical solutions. This can facilitate scientists to produce better models that balance well between the flexibility of learning data-driven models and the existing scientific knowledge. We emphasize that this work is not intended to develop a faster and more efficient surrogate model in place of any existing physical model, but to learn unknown constituents of the PDE used to model the (not yet fully understood) physical processes. The named TCE problem is a representative problem from a broader class where the model structure is only partially known.

In a wider sense, we are interested in environmental problems where fundamental parts of the governing equations (or principles used in their derivation) are accepted as "known truth," but where other parts are uncertain or even unknown, and often treated with assumptions, closures, or other approximations. In our TCE example, it is a sorption isotherm that is most uncertain. Other instances of the same problem class are (a) water retention curves in the Richards equation; (b) capillary pressure and saturation relations, relative permeability and saturation relations, or expressions for hysteresis and dynamic effects in multiphase flow in porous media; or (c) turbulence closures in rivers, pipe flow, or atmospheric flow. When these uncertain or unknown relationships are successfully learned from data, exceptional generalization ability and highly accurate predictions can be expected.

The proposed FINN framework is capable to jointly learn unknown constitutive/closure relationships, PDE terms, and parameters from data. The benefits of our hybrid model are an excellent generalization ability beyond sparse training data, a proper treatment of different boundary conditions other than a constant Dirichlet condition, and an explainable model. Furthermore, with the adoption of the FVM structure and physical constraints, FINN utilizes as much existing modeling knowledge as possible. Additionally, we enable FINN to provide an uncertainty estimate over its learned constituents when predicting a real-world soil contamination problem, and affirm FINN's advantages over a calibrated, conventional PDE-based model.

2. Methodological Background

In this section, we derive the methodological framework of this paper on the basis of an experimental reference setup and the involved equation form which will be learned by FINN. Then, we provide some background on the FVM discretization method to solve PDEs and on neural ODE (NODE) as a differentiable numerical integrator as it serves the conceptual basis for FINN. Finally, we summarize a selection of UQ methods that can be implemented in FINN.

2.1. Experimental Setup and Governing Equations

The diffusion process in general is governed by a PDE of second-order in space and first-order in time:

$$\frac{\partial c}{\partial t} = \nabla \cdot (D(c)\nabla c) + q(c), \tag{1}$$

where c is the variable of interest, namely, the contaminant concentration in this study, t is time, D is a diffusion coefficient that can be a dependent variable on c, and q is the source/sink term, for example, if there is a reaction or there is an addition/extraction of the contaminant to/from the domain of interest. The diffusion through clay, however, might be hampered by the presence of organic matter inside the clay that sorbs the TCE, therefore slowing down the diffusion process (Parker et al., 2004). Therefore, the sorption process has to be taken into account in the governing PDE as well, by including an additional variable in form of the retardation factor. The retardation factor is a variable that is possibly dependent on the contaminant concentration (among other features such as pH, ionic strength, water chemistry), and it defines the degree to which the diffusion process is hindered by the sorption process. The resulting diffusion-sorption equation can be solved with various numerical discretization methods, one of the most popular being FVM due to its conservation property (Moukalled et al., 2016).

This diffusion-sorption process of TCE as contaminant in water-saturated clay was studied in a laboratory experiment (Nowak & Guthke, 2016), and its setup is adopted in the numerical experiments performed in this work. A clay sample with a radius of 2.54 cm (1 inch) is placed inside a stainless steel tube of length L. On the upper end of the sample, pure-phase TCE is injected through an inlet valve. There, it forms a pool, from which it can migrate into the clay and thus installs a constant concentration condition at the upper end of the clay cylinder. The bottom end of the sample is flushed with clean water below the clay cylinder at a constant flow rate, in order to enable measurement of the dissolved TCE concentration (as it diffused downward through the clay) at various time intervals. At the end of the experiment, the clay sample is cut into horizontal slices to allow measurement of the total TCE concentration (i.e., TCE dissolved in the water and sorbed in the clay) within the cylinder. In short, there are two main variables of interest in the experiment, namely the dissolved concentration and the total concentration of the contaminant. More details of the experiment can be found in Parker et al. (2004) and Nowak and Guthke (2016).

Assuming that the clay sample is homogeneous, the governing diffusion-sorption equation can be simplified into a one-dimensional system. Mathematically, the governing PDE used to calculate the dissolved concentration could be written as (Nowak & Guthke, 2016)

$$\frac{\partial c}{\partial t} = \frac{D}{R(c)} \frac{\partial^2 c}{\partial x^2}, \tag{2}$$

where c is the dissolved TCE concentration, t is time, x is distance along the axis of the cylinder, D is the effective diffusion coefficient, and R is the retardation factor, which is a function of c. As a consequence, the diffusivity (i.e., D/R) is also dependent on c.

Because the upper end of the sample is in equilibrium with the TCE in pure-phase, a Dirichlet boundary condition is applied:

$$c|_{x=0} = c_{\text{sol}} \quad \forall t : 0 \le t \le T, \tag{3}$$

where c_{sol} is the solubility limit of the TCE in water, and T is the experiment time. On the bottom end of the sample, the TCE concentration is not constant, and therefore, a Cauchy condition is required to model the flow-dependent boundary condition as a result of the flushing with water:

$$c|_{x=L} = \frac{D}{Q} \frac{\partial c}{\partial x} \quad \forall t : 0 \le t \le T, \tag{4}$$

where Q is the water flow rate at the bottom of the clay sample. The clay sample is initially clean of any contamination, resulting in an initial condition of:

$$c|_{t=0} = 0 \quad \forall x : 0 \le x \le L. \tag{5}$$

To derive a possible equation for the total concentration, the general definition of retardation factor R is required. The retardation factor R is defined as the ratio of sorbed to non-sorbed material as following (e.g., Fetter, 1999; Nowak & Guthke, 2016):

$$R = \frac{1}{\phi} \frac{\partial c_t}{\partial c}, \tag{6}$$

where ϕ is the porosity of the porous medium and c_t is the total contaminant concentration, that is, the contaminant concentration dissolved in the fluid and sorbed in the solid phase. By substituting Equation 6 into Equation 2, the equation to calculate the total contaminant concentration c_t can be written as:

$$\frac{\partial c_t}{\partial t} = D\phi \frac{\partial^2 c}{\partial x^2}. \tag{7}$$

Many sorption processes are well-understood. However, when it involves an interaction between the solvent and sediments, the complexity of the process is enhanced. As a result, parametric models such as the sorption isotherms are commonly assumed for simplification. Three of the most commonly used isotherms are linear,

Freundlich, and Langmuir, which are all derived empirically. These isotherms define the retardation factor differently:

$$R_l = 1 + \frac{1-\phi}{\phi}\rho_s K_d, \tag{8}$$

$$R_F = 1 + \frac{1-\phi}{\phi}\rho_s K_f n_f c^{n_f - 1}, \tag{9}$$

$$R_L = 1 + \frac{1-\phi}{\phi}\rho_s \frac{s_{\max} K}{(c+K)^2}, \tag{10}$$

where Equations 8–10 describe the retardation factor formulation based on the linear, Freundlich, and Langmuir isotherm, respectively. Here, ρ_s is the bulk density of the porous medium, K_d is the linear isotherm parameter, K_f is the Freundlich isotherm parameter, n_f is the Freundlich exponent, $s_{\max}$ is the maximum sorption capacity of the solid phase, and K is the half-saturation value. Note that, if the Freundlich exponent $n_f = 1$, the Freundlich isotherm becomes identical to the linear isotherm, and therefore, it is impossible to distinguish between them.

Traditionally, the retardation factors reported in Equations 8–10 would lead to three different discrete models, one for each sorption isotherm. However, FINN allows us to define the retardation factor as a flexible function that is learned from data to best support the approximation of the overall process without constraining the model too much by a possibly inaccurate assumption (more details on the physical constraint will be discussed in the loss function definition in Section 3). This particular aspect of FINN is demonstrated with the application on laboratory measurement data, which will be discussed in Section 4.2.

2.2. Numerical Solution

Obtaining an analytical solution is impossible for many PDEs. Consequently, it is common to resort to numerical methods to solve it. The most popular numerical methods are the finite difference method (FDM), which approximates the derivatives based on Taylor's expansion (Morton & Mayers, 1994); the finite element method (FEM), which reformulates the PDE in a weak form and interpolates the solution through a function with limited element support (Logan, 1992); and FVM, which approximates the solution using a volume integral combined with the Gauss' divergence theorem (Moukalled et al., 2016).

The FVM derivation of the PDE is based on conservation laws and is the closest to physics compared to the other discretization methods. To be more specific, the applied divergence theorem leads to the right-hand side of Equation 13, which now represents the flux exchanges between any control volume and its neighboring volumes. This ensures that conservation is not violated, meaning that the flux entering a control volume should be exactly the same as the flux leaving the control volume, given that the variable of interest (i.e., the concentration or quantity c) does not change over time. On the other hand, FEM does not guarantee this conservation property. Moreover, FVM allows straightforward implementation of the boundary conditions without approximation. Due to these reasons, we choose to specifically adopt FVM in this work. Note that, when the domain is discretized using a regular Cartesian grid, then the FVM, FEM, and FDM discretizations are identical, with differences in the boundary condition treatment.

Following the FVM concept, the spatial domain is discretized into a number of N control volumes (cells). For each control volume, a volume integral is applied to Equation 2, resulting in

$$\int_{v_i} \frac{\partial c}{\partial t} dv = \int_{v_i} \frac{D}{R(c)} \frac{\partial^2 c}{\partial x^2} dv, \tag{11}$$

where v_i is the volume, and the subscript $i = 1, \ldots, N$ denotes a specific control volume i. Since the right-hand side has a divergence term, the Gauss' divergence theorem needs to be applied (Arfken et al., 2013), leading to a surface integral over the enclosing control volume surfaces/boundaries, and resulting in the equation

$$\int_{v_i} \frac{\partial c}{\partial t} dv = \oint_{\omega \subseteq \Omega} \left(\frac{D}{R(c)} \frac{\partial^2 u}{\partial x^2} \right) \cdot \hat{n}\, d\Gamma, \tag{12}$$

where ω is a continuous surface element and $\imath$ is the unit normal vector pointing outwards of ω. Furthermore, ω is a subset of all surfaces Ω enclosing the control volume i and Γ is a continuous variable along ω. Applying the surface integral in Equation 12 allows flux evaluation at each enclosing control volume surface. As a result, a spatially discrete formulation of the PDE for control volume i using a Cartesian grid leads to the following:

$$\frac{\partial c_i}{\partial t} v_i = A_{i-1} \frac{D_i}{R(c_i)} \frac{c_{i-1} - c_i}{\Delta x} - A_{i+1} \frac{D_i}{R(c_i)} \frac{c_i - c_{i+1}}{\Delta x}, \tag{13}$$

where A_{i-1} and A_{i+1} are the left and right cross-sectional surface areas of control volume i, respectively. The equation is spatially discrete, and thus uses Δx instead, which is the length of the control volume.

For each $i = 1, \ldots, N$, Equation 13 form a coupled set of ODEs. The ODEs in the system of Equation 13 are coupled through their connections with their respective neighbors ($i - 1$ and $i + 1$). To derive Equation 13 into a spatially and temporally discrete equation, a temporal discretization is required. The simplest temporal discretization scheme is the Euler method (e.g., Butcher, 2008), which is a first-order time integration method. The Euler method itself is categorized into explicit and implicit schemes. In the explicit scheme, the time derivative function dc/dt is defined with the variable c at the current time step t, whereas in the implicit scheme, it is defined with the variable c at the subsequent time step $t + 1$. Applying the explicit Euler method to Equation 13 leads to

$$\frac{c_i^{t+1} - c_i^t}{\Delta t} v_i = A_{i-1} \frac{D_i}{R\left(c_i^t\right)} \frac{c_{i-i}^t - c_i^t}{\Delta x} - A_{i+1} \frac{D_i}{R\left(c_i^t\right)} \frac{c_i^t - c_{i+1}^t}{\Delta x}, \tag{14}$$

and applying the implicit Euler method yields

$$\frac{c_i^{t+1} - c_i^t}{\Delta t} v_i = A_{i-1} \frac{D_i}{R\left(c_i^{t+1}\right)} \frac{c_{i-i}^{t+1} - c_i^{t+1}}{\Delta x} - A_{i+1} \frac{D_i}{R\left(c_i^{t+1}\right)} \frac{c_i^{t+1} - c_{i+1}^{t+1}}{\Delta x}, \tag{15}$$

where the superscript t denotes the time discretization and Δt is the corresponding time step. The same discretization strategy also applies to Equation 7. As can be inferred from both equations, the implementation of the explicit method is simpler than the implicit method, because the value of c_i^{t+1} is still unknown in time step t. Furthermore, because we intend to combine numerical methods with ANNs—which fundamentally belong to the class of explicit methods—from here on we will use the explicit scheme and, when possible, drop the superscript t for clarity. More comprehensive explanations on links between ANNs and differential equations can be found in (K. He et al., 2016; Y. Lu et al., 2018), where the authors discuss the analogy between the skip connections in Residual Networks (ResNet) and a numerical discretization method, as well as neural operator learning (Kovachki et al., 2021).

Even though the explicit method is more convenient to implement, it suffers from numerical instability. To be more specific, the size of the time step Δt has to be chosen carefully such that it does not surpass the time at which the quantity c is moving from one control volume to the other. This requirement can be controlled, for example, by using the Courant–Friedrichs–Lewy (CFL) condition (Courant et al., 1967; Isaacson & Keller, 1994). According to CFL, a finer spatial discretization (i.e., smaller Δx) requires a smaller temporal discretization Δt. In some cases, this condition becomes very limiting when the required Δt becomes too small, leading to substantially inefficient computation. As an alternative, a higher order integration method can be used to alleviate this limitation, which will be discussed in the following section.

2.3. Neural ODE

Implementing the explicit integration method from Section 2.2 with an unregularized ANN will likely lead to numerical instability, as the ANN might easily enter unstable regimes. To mitigate this problem, a higher-order ODE integration method, such as Runge-Kutta method (Kutta, 1901; Runge, 1895), can be used, in combination with an adaptive time stepping capability, to maintain both the numerical stability and the accuracy of the integration. Such ODE solvers, however, must be differentiable in order to be able to propagate an error signal for adapting the parameters of an ANN, which will be used to represent the ODE. For a comprehensive introduction to ANNs, we refer the readers to Goodfellow et al. (2016).

Such a *fully differentiable* and ANN-based ODE solver, called NODE, has recently been proposed by Chen et al. (2018). In short, NODE assumes an ANN f_θ with parameters θ to compute the change of a state vector $\mathbf{c}^t$ over time. NODE parameterizes the change of $\mathbf{c}$ over time by treating f_θ as a time-continuous function, such that

$$\frac{\partial \mathbf{c}(t)}{\partial t} = f_\theta(\mathbf{c}(t), t). \tag{16}$$

Accordingly, $f(\cdot)$ is an ANN that learns and represents the system dynamics, that is, the derivative of the variable of interest with respect to time, which directly corresponds to the form of Equation 13. Then, the quantity of $\mathbf{c}$ in the next time step could, in principle, be computed using an explicit Euler discretization:

$$\mathbf{c}^{t+1} = \mathbf{c}^t + f_\theta\left(\mathbf{c}^t\right). \tag{17}$$

Formulating a dynamic system with Equation 17 has been proposed in the Deep Residual Learning (ResNet) architecture (K. He et al., 2016). This approach has been shown to improve model training because it can learn particular functions, such as the identity function, better and because it minimizes the vanishing gradient problem (Hochreiter et al., 2001). In the NODE framework, however, the temporal discretization of Equation 16 occurs after the forward propagation of the ANN and with a higher-order scheme, rather than learning the discretization form as in Equation 17.

In NODE, to integrate from $\mathbf{c}^t$ to $\mathbf{c}^{t+1}$, f_θ is optimized end-to-end in the overall training process. This leads to better accuracy and efficiency, as well as to allow adaptive time stepping strategy for better numerical stability (Chen et al., 2018). Note that this approach differs from the Elman network, that is, the traditional recurrent neural network (Elman, 1990). The Elman network uses the ANN as a function to predict $\mathbf{c}^{t+1}$ directly from $\mathbf{c}^t$, that is, without the appearance of $\mathbf{c}^t$ in Equation 17. NODE plays a fundamental role in our model, as detailed in Section 3.

2.4. Uncertainty Quantification Methods

In this work, we perform UQ over the model parameters and all learned constituents, such as the retardation factor function $R(c)$. One of the most straightforward UQ methods on ANNs is the Bayes-by-backprop method (Blundell et al., 2015), which parameterizes the variational posterior using the mean μ and standard deviation σ of the model parameters θ (i.e., weights and biases). In short, for each training iteration, the model parameters are sampled based on

$$\theta = \mu + \log(1 + \exp(\sigma)) \cdot \epsilon, \tag{18}$$

where $\epsilon \sim \mathcal{N}(0, I)$. The goal of the training is then to find the values of μ and σ that minimize the Kullback-Leibler divergence (Joyce, 2011) between the variational posterior $q(\theta)$ and the (unknown) true posterior $\pi(\theta)$, reformulated as

$$\mathcal{L} = KL[q(\theta)\|p(\theta)] - \mathbb{E}_{q(\theta)}[\log p(D|\theta)], \tag{19}$$

where $p(\theta)$ is the prior knowledge on the model parameters, and $p(D|\theta)$ is the probability of observing the data D given the model parameters θ.

The Bayes-by-backprop approach, however, assumes independent Gaussian distributions to define the model parameters, which is an oversimplification of the actual joint posterior distribution (Blei et al., 2017). In contrast, Markov chain Monte Carlo (MCMC; Bardenet et al., 2017) provides a sampling of model parameters from the exact posterior distribution (Jospin et al., 2022). The general MCMC algorithm is summarized in Algorithm 1.

The proposal of drawing θ_t and the transition/proposal distribution Q are defined respectively for the random walk Metropolis-Hastings (MH; Chib & Greenberg, 1995), Metropolis-adjusted Langevin algorithm (MALA; Dwivedi et al., 2019), and Barker proposal (Barker, Livingstone & Zanella, 2019), as:

$$\theta_t = \theta^{(i)} + h \cdot \epsilon, \qquad Q\left(\theta_t|\theta^{(i)}\right) = \exp\left(-\|\theta_t - \theta^{(i)}\|_2^2/h^2\right), \tag{20}$$

$$\theta_t = \theta^{(i)} + h\nabla\pi\left(\theta^{(i)}\right) + \sqrt{2h}\epsilon, \qquad Q\left(\theta_t|\theta^{(i)}\right) = \exp\left(-\|\theta_t - \theta^{(i)} - h\nabla\pi\left(\theta^{(i)}\right)\|_2^2/4h\right), \tag{21}$$

Algorithm 1. General Markov Chain Monte Carlo Algorithm

Require: Initial parameter values $\theta^{(0)}$
  Set $i = 0$
  while $i < N$ **do**
    Draw θ_t given $\theta^{(i)}$
    Calculate acceptance probability $\alpha\left(\theta_t|\theta^{(i)}\right) = \min\left(1, \frac{\pi(\theta_t)Q(\theta^{(i)}|\theta_t)}{\pi(\theta^{(i)})Q(\theta_t|\theta^{(i)})}\right)$
    Draw a random number $u \sim \mathcal{U}[0,1]$
    if $\alpha(\theta_t|\theta^{(i)}) > u$ **then**
      $\theta^{(i+1)} \leftarrow \theta_t$
    else
      $\theta^{(i+1)} \leftarrow \theta^{(i)}$
    end if
    $i \leftarrow i + 1$
  end while

$$\theta_t = \theta^{(i)} + b \cdot \epsilon, \qquad Q\left(\theta_t|\theta^{(i)}\right) = \frac{1}{1 + \exp\left(-(\theta_t - \theta^{(i)})^T \nabla \log \pi\left(\theta^{(i)}\right)\right)}. \tag{22}$$

Here, θ_t is the proposed sample, $\theta^{(i)}$ is the sample from iteration i, h is the step size, π is the posterior of the model parameters, and $\epsilon \sim \mathcal{N}(0, I)$. For Barker specifically, $b = 1$ with the probability $p = 1/(1 + \exp(-\epsilon \nabla \pi(\theta^{(i)})))$, and $b = -1$ otherwise. It is also important to note that both MALA and Barker utilize the gradient information provided by the automatic differentiation tools available in various ML libraries, including PyTorch (Paszke et al., 2019), which is used in this work.

3. Finite Volume Neural Network

In this section, we introduce the FINN framework by providing derivations and explanations on how FINN relates to the concepts from the previous section. FINN is designed to explicitly learn the individual components of the PDE using dedicated modules—defined as nonlinear ANN layers—in a compositional manner (Battaglia et al., 2018; Karlbauer et al., 2022; Lake, 2019; Lake et al., 2017). These modules are connected to effectively represent the PDE of interest (see Figure 1). Although the model has a general form, it can be set up and interconnected individually to form a specific architecture that is explicitly motivated and inspired by the physical

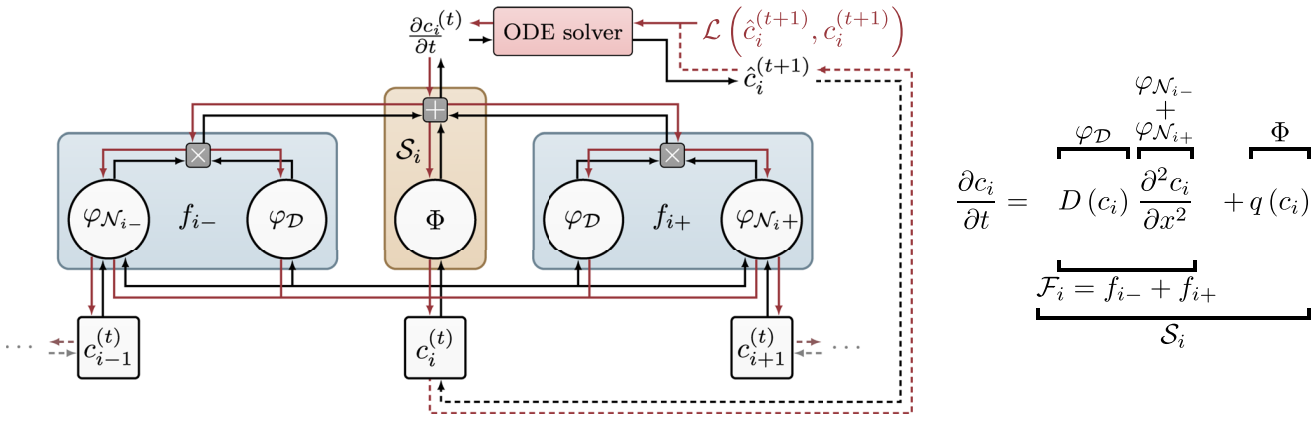


Figure 1. Schematic illustration of a flux kernel for one finite volume in finite volume neural network (left) and alignment of network modules with their corresponding parts in a partial differential equation of interest (right). Black lines indicate forward information flow whereas red lines indicate gradient flow during backpropagation through time. Dashed lines indicate closed-loop feedback between subsequent time steps.

equation that would typically be assumed to govern the modeled system at hand, that is, the diffusion-sorption Equation 2 in this work. More specifically, FINN combines the knowledge-based structure of the PDE core elements with the FVM discretization as described in Section 2.2 to obtain a set of ODEs. Then, it adopts the NODE method as described in Section 2.3 for the time integration and learning of the PDE constituents (i.e., not only the PDE parameters, but also unknown functions such as the retardation factor and the reaction function, as well as the discretization of the PDE).

As outlined in Section 2.2, a PDE describes the change of a quantity at a local position under influence of its direct neighbors. Accordingly, we propose to model the adjacent flux exchange—see Equation 13—by so called flux kernels $\mathcal{F}_i$. They learn to represent the quantity entering and leaving control volume i from left and right (in the one-dimensional case). These flux kernels approximate the surface integral for each control volume i as written in Equation 12, and therefore are mathematically written as

$$\mathcal{F}_i = \sum_{j=1}^{M_i} f_j \approx \oint_{\omega \subseteq \Omega} \left(D(c) \frac{\partial^2 c}{\partial x^2} \right) \cdot \hat{n} \, d\gamma \tag{23}$$

where M_i is the number of discrete surface elements of control volume i and f_j is the subkernel calculated at each surface element. For one-dimensional cases, $M_i = 2$ and therefore, each flux kernel is supported by 2 subkernels $\mathcal{F}_i = \{f_{i-}, f_{i+}\}$.

The flux kernels $\mathcal{F}_i$ are provided with c_i and c_{i-1} or c_{i+1} as the inputs for f_{i-} or f_{i+}, respectively. Each subkernel consists of two modules that are formulated as neural network layers. On the one hand, $\varphi_{\mathcal{N}}$ is a linear layer to approximate the FVM stencil to represent the contribution of each neighboring control volume and the direction of the flux exchange. Hence, the output of $\varphi_{\mathcal{N}}$ is conceptually supposed to become $\partial^2 c / \partial x^2$, that is

$$\varphi_{\mathcal{N}_{i-}}(c_i, c_{i-1}) + \varphi_{\mathcal{N}_{i+}}(c_i, c_{i+1}) \approx \frac{\partial^2 c_i}{\partial x^2}. \tag{24}$$

Note that $\varphi_{\mathcal{N}_{i-}}$ and $\varphi_{\mathcal{N}_{i+}}$ share weights and thus are represented by one and the same network. Ideally, in a system with Fickian diffusion and mass conservation fulfilled, the parameters of $\varphi_{\mathcal{N}}$ should be $[-1, 1]$ with respect to $[c_i, c_{i-1}]$ and $[c_i, c_{i+1}]$. When f_{i-} and f_{i+} are combined, that is, in Equation 24, the coefficients become $[1, -2, 1]$ with respect to $[c_{i-1}, c_i, c_{i+1}]$. This follows from the central discretization scheme (Fornberg, 1988) of the second-order spatial derivative as

$$\frac{\partial^2 c}{\partial x^2} \approx \frac{(\partial c / \partial x)|_{i-} - (\partial c / \partial x)|_{i+}}{\Delta x}, \tag{25}$$

with $(\partial c / \partial x)|_{i-} \approx (c_{i-1} - c_i)/\Delta x$ and $(\partial c / \partial x)|_{i+} \approx (c_i - c_{i+1})/\Delta x$. As a result,

$$\frac{\partial^2 c}{\partial x^2} \approx \frac{(c_{i-1} - c_i) - (c_i - c_{i+1})}{\Delta x^2} = \frac{c_{i-1} - 2c_i + c_{i+1}}{\Delta x^2}. \tag{26}$$

The module φ_D, on the other hand, is responsible to account for the (variable-dependent, possibly nonlinear) diffusion coefficient, thus

$$\varphi_D(c_i) \approx D(c_i), \tag{27}$$

if the diffusion coefficient D depends on c. Otherwise, φ_D is a scalar value $\varphi_D \equiv D$, which can also be set as a learnable parameter. For our exemplary diffusion-sorption case, $\varphi_D \approx D/R(c)$ for the dissolved concentration, that is, Equation 2, and $\varphi_D \approx D$ for the total concentration, that is, Equation 7. Because the diffusion coefficient D can be learned from the total concentration, the retardation factor $R(c)$ can also be extracted from the learned module φ_D in the dissolved concentration calculation.

Finally, subkernels f_{i-} and f_{i+} are calculated as a combination of both modules $\varphi_{\mathcal{N}}$ and φ_D:

$$f_{i-} = \varphi_D(c_i) \cdot \varphi_{\mathcal{N}_{i-}}(c_i, c_{i-1}), \tag{28}$$

$$f_{i+} = \varphi_D(c_i) \cdot \varphi_{\mathcal{N}_{i+}}(c_i, c_{i+1}). \tag{29}$$

Performing calculations of flux kernels at the surface element additionally provides advantages for boundary condition treatment. In addition to Dirichlet, boundary condition types that are flux dependent, such as Neumann or Cauchy conditions (Cheng & Cheng, 2005), can easily be adopted. For Dirichlet boundary conditions, a constant value $c = c_b$ is set as the input c_{i-1} (for the flux kernel f_{i-}) or c_{i+1} (for f_{i+}) at the corresponding boundaries. For a Neumann boundary condition ν, the output of the flux kernel f_{i-} or f_{i+} at the corresponding boundaries can be set to be equal to ν. For Cauchy boundary conditions, the solution-dependent derivative is calculated and set as c_{i-1} or c_{i+1} at the corresponding boundary.

Furthermore, we also introduce the state kernel $\mathcal{S}_i$ to model $\partial c/\partial t$. The state kernel $\mathcal{S}_i$ receives c_i of the associated control volume i, along with the output of f_{i-} and f_{i+} (the fluxes to/from each neighboring cell) as inputs:

$$\mathcal{S}_i\left(c_i, f_{i-}, f_{i+}\right) = \left(\Phi\left(c_i\right) + f_{i-} + f_{i+}\right) \approx \frac{\partial c_i}{\partial t}. \tag{30}$$

Optionally, the state kernel includes a neural network module $\Phi(\cdot)$, a function that is trained to model reaction terms related to the quantity c_i in control volume i. In the processes considered in this work, however, the only source of change comes from neighboring volumes. Particularly, there are no external effects that locally modify the quantity of interest (such as e.g., sun radiation would increase temperature locally without sensible or latent heat entering from adjacent volumes). Therefore, in this work, $\Phi(\cdot)$ is the identity function and, hence, $\mathcal{S}_i$ is only responsible for integrating information coming from neighboring cells.

Taking the bigger picture into account, all control volumes share the same kernels, and thus FINN is parsimonious, exploiting translation equivariance of physical laws. Flux kernels specifically are designed similarly to message passing neural networks (Brandstetter et al., 2022; Gilmer et al., 2017) to exploit PDE-type structural knowledge upon discretization. The main difference lies in the fact that all flux kernels are uniquely labeled with a physical meaning through derivation from FVM and NODE. Additionally, boundary conditions can be switched explicitly.

During training, FINN only receives the initial condition $c(x, t = 0)$ as input. This input is processed first by the flux kernels to calculate the flux exchanges between neighboring control volumes, and then by the state kernels to be integrated through all surface elements. The output of the state kernels are then fed into a differentiable ODE solver (within a NODE framework) to be integrated in time to obtain the solution c at the subsequent time step. This solution is fed back into FINN, and the same operations are recurrently applied until the final simulation time $t = T$ is reached. This way, FINN uses a closed-loop setting to propagate the dynamics forward, leading to a more stable prediction during testing (Praditia et al., 2020). This workflow is visualized in Figure 1, where the black arrows depict the direction of the input processing, and the red arrows depict the direction of the error backpropagation during training. The dashed arrows depict the closed-loop feedback between subsequent time steps, as well as coupling between neighboring control volumes.

The prior physical information in FINN is embedded in the form of the additional physical regularization and the structure itself, not in the training algorithm. For this particular case, FINN is trained by minimizing the loss function, defined by combining the data driven error and physical constraint:

$$\mathcal{L} = \frac{1}{N_e}\sum_{i}^{N_e}\left(c_i - \hat{c}_i\right)^2 + \frac{1}{N_p}\sum_{j}^{N_p} E_{p,j}, \tag{31}$$

where c is the available training data (either generated from simulation or measured from experiment), $\hat{c}$ is FINN's prediction, N_e is the number of data points of $c(x, t)$, E_p is an additional physical constraint, and N_p is the number of data points used to calculate the physical constraint. The physical constraint E_p in our current example enforces that the retardation factor is a monotonically decreasing function of c, that is, $R(c_i) \geq R(c_j)$ for all $c_i < c_j$ and positiveness of the diffusion coefficient, that is, $D > 0$. To enforce these conditions, the ReLU operator is used. Other thinkable constraints could ab initio dictate mass conservation, energy balances or thermodynamic principles. Likewise, soft information (like staying close to an often-successful but not fully accurate law) could also be added. This loss function is also used as the negative log posterior $-\pi(\theta)$ in the UQ, which will be discussed later.

Table 1
Parameter Values for Synthetic Data Generation

Parameter	Symbol	Unit	Value
Common parameters			
Effective diffusion coefficient	D	m^2/day	5.00×10^{-4}
Porosity	ϕ	–	0.29
Density	ρ_s	kg/m^3	2,880
Linear isotherm			
Partitioning coefficient	K_d	m^3/kg	4.30×10^{-4}
Freundlich isotherm			
Freundlich's K	K_f	$(m^3/kg)^{n_f}$	3.50×10^{-4}
Freundlich exponent	n_f	–	0.87
Langmuir isotherm			
Half-concentration	K	kg/m^3	1.00
Sorption capacity	s_{max}	m^3/kg	5.90×10^{-4}

4. Learning Experiments

To demonstrate FINN's capability, both synthetic and real-world data sets of the TCE diffusion-sorption process are used. Recall that the hypothesized advantages of FINN, by construction and due to the physics constraints, are little data requirements, robust generalization beyond the training data distribution (i.e., applicability to different boundary conditions), and high interpretability, allowing it to learn when known models fail.

First, we generate synthetic data based on the setup described in Section 2.1 in order to perform an elaborated analysis in a controlled setting. By doing so, we will also assess the performance of FINN against already existing pure deep learning models such as TCN and ConvLSTM, physics-motivated deep learning models, such as DISTANA and FNO, as well as physics-aware architectures, that is, PINN and PhyDNet. Second, we also show that FINN is not only suitable for synthetic data, but also for real-world application by modeling real laboratory experimental data.

4.1. Synthetic Data Set

We simulate the experimental setup described in Section 2.1 numerically and generate synthetic data sets solving the related (assumed-to-be-true for now) PDE. The numerical simulator is a simple finite difference code with explicit Euler, made available along with our code (Praditia et al., 2022). Three different sorption isotherms, namely the linear, Freundlich, and Langmuir isotherm are used to generate three distinct synthetic data sets, which are then used for a comparison study. The parameters for each isotherm are set so that they yield similar concentration distribution, thus we can show that our proposed method is able to distinguish various isotherms even with similar looking data (only by approximating the function using an ANN, and not explicitly choosing between the available equations). The parameter values are given in Table 1.

To compare the generalization ability of FINN to that of the aforementioned existing methods, for each data generated with different sorption isotherms, we define three different types of synthetic data sets: *train*, in-distribution test (*in-dis-test*), and out-of-distribution test (*out-dis-test*). The differences between these data sets lie in the time domain and boundary condition used. The *train* data is generated with $x = [0, 1]$ m, $t = [0, 2{,}500]$ days, and $c_{sol} = 1.0$ kg/m^3. The *in-dis-test* data is generated with $x = [0, 1]$ m, $t = [2{,}500, 10{,}000]$ days, and $c_{sol} = 1.0$ kg/m^3. The *out-dis-test* data is generated with $x = [0, 1]$ m, $t = [0, 10{,}000]$ days, and a modified upper boundary value of $c_{sol} = 0.7$ kg/m^3. The simulation domain for all three types of data is discretized with $\Delta x = 0.04$ m and $\Delta t = 5$ days. The *train* data, as its name suggests, is used to train the models. Both *in-dis-test* and *out-dis-test* are used to test the models. The difference is that *in-dis-test* data is generated with the same parameter as the *train* data, but extrapolated for a longer time span, whereas *out-dis-test* data is generated with a different boundary condition value, to test the models' generalization under a different situation than during training. A different *type* of boundary condition will be tested in Section 4.2.

4.1.1. State-of-the-Art Benchmark Models

In this work, we compare FINN's performance against other models that are capable of processing spatiotemporal data. These models are either pure ML models or models that possess a form of physical inductive bias.

Pure ML models. For the pure ML models, we choose **TCN** and **ConvLSTM**. **TCN** performs convolution operations over both the spatial and temporal domain (Lea et al., 2016), and it exploits the benefits of features such as dilated convolution to process a larger receptive field. In other words, the **TCN** structure allows for processing information contained in more distant preceding time steps. For **TCN**, the chosen structure has 2 input channels, a hidden layer with 32 channels, and 2 output channels.

ConvLSTM takes a more classical approach, which is to capitalize on the recurrent structure of the long short-term memory (LSTM) model to handle the temporal correlation of the data, and replaces the internal operations with

convolutional operations to handle the spatial correlation of the data (Shi et al., 2015). For **ConvLSTM**, the chosen structure is 2 input and output channels, with a hidden layer containing 24 channels.

Physics-motivated ML models. The physics-motivated methods chosen as benchmark models in this work are namely **DISTANA**, **FNO**, and **CNN-NODE**. While **DISTANA** is similar to **ConvLSTM**, the main difference is that **DISTANA** propagates information laterally via additional latent feature maps (and not only by applying convolutions on the input). This can also be seen as an analog to a flux exchange between neighboring control volumes—even though in **DISTANA**, the lateral latent information does not have any physical meaning. The lateral and dynamic input and output sizes of the **DISTANA** model are set to 1 and 2, respectively, while a hidden layer of size 16 is used.

CNN-NODE is a combination of a conventional convolutional network stem that is augmented by NODE and thus formulates an ablation of FINN to determine the relevance of FINN's modular network architecture. We use a three-layered, batch-normalized and tanh-activated convolution stem.

Due to a point-wise formulation (similarly to PINN), **FNO** approximates PDEs by learning a continuous mapping from space-time inputs to the desired outputs, that is, $\mathbb{R}^{x\times t}\mapsto\mathbb{R}^{d}$, where x and t are space and time coordinates and d is the dimensionality of the target. However, FNO differs from PINN in two fundamental aspects: First, FNO learns the according mapping in frequency instead of in time domain by applying fast (inverse) Fourier transformations. Second, FNO learns purely from data and does not depend on explicit physical process knowledge. In our experiments, we apply the identical model architecture as suggested by Z. Li et al. (2020a).

Physics-aware ML models. In the class of physics-aware ML models, we chose **PINN** and **PhyDNet** as benchmark candidates. **PINN** is one of the pioneering physics-motivated ML models. It makes use of the capability of ANN to calculate analytical derivatives through backpropagation to approximate derivatives in the PDE (Raissi et al., 2019). For this problem, **PINN** is defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 2] (i.e., 2 input and output neurons, with 8 hidden layers, each containing 20 neurons).

PhyDNet consists of two main branches: one for calculating the physical component and the other for calculating the residual component of the data, assuming that the PDE does not fully describe the modeled system. The physical branch is inspired by the Kalman Filter, which is a data assimilation technique to recurrently update the model parameters based on observation (i.e., training) data. The residual branch adopts the **ConvLSTM** structure. With a specific condition of the **PhyDNet** structure, it reduces to a **PDE-Net** model (Guen & Thome, 2020). **PhyDNet** is defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer.

In the last class of physics-aware ML models, **FINN** is implemented with the use of the modules $\varphi_{\mathcal{N}}$ and φ_{D}. Here, the module $\varphi_{\mathcal{N}}$ is defined as a linear layer that takes 2 inputs, namely the dissolved concentration c of two neighboring control volumes. For the dissolved concentration c, the module φ_{D} is defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes c as an input and outputs the retardation factor $R(c)$. For the total concentration c_t, φ_{D} is defined as a scalar parameter to learn the unknown diffusion coefficient D.

4.1.2. Benchmark Performance of ML Models

All models are trained with the objective to minimize the deviation between the model predictions of c and c_t with the training data. They are trained until convergence using the L-BFGS optimizer (Malouf, 2002), except for PhyDNet and FNO, which are trained with the Adam optimizer (Kingma & Ba, 2015) and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer. The L-BFGS optimizer is chosen because it is a quasi-Newton optimization algorithm, which means it uses an approximation of the second-order derivative (Hessian matrix). Second-order optimization algorithms are shown to be more effective in reaching the (local) optima (Kochenderfer & Wheeler, 2019). The pure ML models are trained on the first 400 time steps ($t=0$ to 2,000) and validated on the remaining 100 time steps ($t=2{,}000$–$2{,}500$) of the train data, applying early stopping (Goodfellow et al., 2016). Additionally, all models are trained with 10 different random initializations to learn about their consistency and to show better representation of each model's performance.

Table 2 shows the summary of all the trained models' performance. For each data set and each model, the mean and standard deviation values of the prediction mean squared error (MSE) across the 10 different initializations are presented. A more detailed presentation of the MSE values for each random training initialization can be found in Supporting Information S1. Note that PINN is not implemented for *out-dis-test* data. The reason behind

Table 2
Comparison of Mean Squared Error and According Standard Deviation Scores Across 10 Repetitions Between Different Deep Learning (TCN and ConvLSTM), Physics-Motivated (DISTANA, CNN-NODE, and FNO), and Physics-Aware Neural Networks (PINN, PhyDNet, and FINN) Methods on the Different Isotherms

			Data set		
Iso.		Model	*Train*	*In-dis-test*	*Out-dis-test*
Linear	Pure ML	TCN	$(2.4 \pm 3.1) \times 10^{-1}$	$(3.5 \pm 4.4) \times 10^{-1}$	$(2.9 \pm 3.1) \times 10^{-1}$
		ConvLSTM	$(3.7 \pm 3.9) \times 10^{-2}$	$(4.0 \pm 3.9) \times 10^{-2}$	$(5.3 \pm 4.9) \times 10^{-2}$
	Physics motiv.	DISTANA	$(2.8 \pm 6.5) \times 10^{-4}$	$(1.9 \pm 2.6) \times 10^{-3}$	$(3.9 \pm 2.3) \times 10^{-3}$
		CNN-NODE	$(2.1 \pm 3.2) \times 10^{-3}$	$(1.6 \pm 1.9) \times 10^{-1}$	$(1.5 \pm 1.8) \times 10^{-1}$
		FNO	$(7.6 \pm 3.3) \times 10^{-5}$	$(1.0 \pm 0.3) \times 10^{-3}$	$(1.9 \pm 0.4) \times 10^{-2}$
	Physics aware	PINN	$(6.3 \pm 11) \times 10^{-5}$	$(3.9 \pm 7.8) \times 10^{-3}$	–
		PhyDNet	$(3.3 \pm 1.5) \times 10^{-5}$	$(6.1 \pm 17) \times 10^{-3}$	$(1.6 \pm 1.0) \times 10^{-2}$
		FINN	$\mathbf{(2.1 \pm 1.5) \times 10^{-7}}$	$\mathbf{(2.7 \pm 1.9) \times 10^{-7}}$	$\mathbf{(1.8 \pm 1.3) \times 10^{-7}}$
Freundlich	Pure ML	TCN	$(1.1 \pm 3.5) \times 10^{-1}$	$(1.6 \pm 1.5) \times 10^{-1}$	$(1.3 \pm 1.3) \times 10^{-1}$
		ConvLSTM	$(1.9 \pm 2.8) \times 10^{-2}$	$(2.4 \pm 1.9) \times 10^{-2}$	$(4.3 \pm 3.8) \times 10^{-2}$
	Physics motiv.	DISTANA	$(8.1 \pm 7.0) \times 10^{-6}$	$(1.8 \pm 1.6) \times 10^{-4}$	$(1.5 \pm 1.4) \times 10^{-3}$
		CNN-NODE	$(4.3 \pm 8.9) \times 10^{-3}$	$(2.6 \pm 5.2) \times 10^{-1}$	$(2.2 \pm 4.3) \times 10^{-1}$
		FNO	$(6.7 \pm 17.6) \times 10^{-4}$	$(1.4 \pm 2.7) \times 10^{-3}$	$(1.4 \pm 0.5) \times 10^{-2}$
	Physics aware	PINN	$\mathbf{(4.3 \pm 2.4) \times 10^{-6}}$	$(9.7 \pm 16) \times 10^{-4}$	–
		PhyDNet	$(7.1 \pm 20) \times 10^{-4}$	$(2.2 \pm 3.7) \times 10^{-3}$	$(1.2 \pm 0.1) \times 10^{-2}$
		FINN	$(2.9 \pm 0.4) \times 10^{-5}$	$\mathbf{(2.7 \pm 0.4) \times 10^{-5}}$	$\mathbf{(2.3 \pm 0.3) \times 10^{-5}}$
Langmuir	Pure ML	TCN	$(1.3 \pm 0.6) \times 10^{-1}$	$(1.2 \pm 0.7) \times 10^{-1}$	$(1.5 \pm 0.5) \times 10^{-1}$
		ConvLSTM	$(3.9 \pm 3.4) \times 10^{-2}$	$(3.1 \pm 2.3) \times 10^{-2}$	$(6.2 \pm 4.4) \times 10^{-2}$
	Physics motiv.	DISTANA	$(2.3 \pm 2.6) \times 10^{-5}$	$(9.8 \pm 14) \times 10^{-4}$	$(3.3 \pm 3.5) \times 10^{-3}$
		CNN-NODE	$(1.8 \pm 3.1) \times 10^{-4}$	$(1.2 \pm 1.2) \times 10^{-1}$	$(9.7 \pm 10.7) \times 10^{-2}$
		FNO	$(3.5 \pm 7.2) \times 10^{-4}$	$(1.1 \pm 1.4) \times 10^{-3}$	$(1.7 \pm 0.7) \times 10^{-2}$
	Physics aware	PINN	$\mathbf{(3.3 \pm 8.9) \times 10^{-5}}$	$(6.4 \pm 17) \times 10^{-3}$	–
		PhyDNet	$(4.6 \pm 4.2) \times 10^{-5}$	$(1.3 \pm 1.4) \times 10^{-3}$	$(1.3 \pm 0.2) \times 10^{-2}$
		FINN	$(7.3 \pm 7.2) \times 10^{-5}$	$\mathbf{(7.9 \pm 7.8) \times 10^{-5}}$	$\mathbf{(6.1 \pm 6.1) \times 10^{-5}}$

Note. Best results are reported in bold.

this is that PINN learns the explicit relationship of the prediction as a function of x and t based on a specific initial and boundary condition implemented in the *train* data. When the boundary condition is changed to generate the *out-dis-test* data, this functional relationship no longer holds, and as such, PINN is no longer applicable. As a further comparison, we present the number of learnable parameters used by the different models in Table 3. FINN's parameters consist of two parameters of the $\varphi_{\mathcal{N}}$ module as a linear layer, and the weights and biases of the φ_D module as a feedforward ANN. With a well-defined structure, FINN manages to use a relatively small number of parameters to achieve better results in comparison to the other models. To clarify, the benchmark does not include computational time, since the main purpose of our work is not to build a fast surrogate model, but

Table 3
Comparison of the Number of Learnable Parameters Used by the Different Models

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
10,782	8,216	6,519	1,026	5,878	3,042	37,815	**464**

Note. Each model uses the same number of parameters for the linear, Freundlich, and Langmuir cases. FINN uses the least number of parameters (printed in bold).

rather to learn unknown functions in an interpretable fashion from the data and to generalize well to unseen data with different initial and boundary conditions.

As shown in Table 2, the pure ML models perform very poorly even during training, failing to capture/approximate the system's behavior. This result is expected because we only use a relatively small amount of training data, whereas pure ML models thrive on learning from large amount of data. However, measurement data on physical systems are often not abundantly available. Therefore, it is important to emphasize the capability of the models to learn from limited data. The physics-motivated and physics-aware models, on the other hand, perform comparably during training, showing adequate learning (except for CNN-NODE). Also, note that the pure ML models in general require more parameters compared to the physics-motivated and physics-aware models, as shown in Table 3. Even then, their performance is still not comparable. However, PhyDNet is an exception, because most of its parameters originate from the ConvLSTM branch, which is the data-driven part and not the physics-aware part of PhyDNet. During training, FINN achieves the lowest prediction error for the data generated with the linear sorption isotherm, while PINN achieves the lowest prediction error for the data generated with the other sorption isotherms, namely Freundlich and Langmuir. This relatively higher training error of FINN on the Freundlich and Langmuir isotherm data can be attributed to the error of the diffusion coefficient prediction, which will be discussed further in the following section. We also note that the training times of the physics-motivated and physics-aware models are comparable to the pure ML models, without providing a rigorous computational time comparison.

It is more interesting to see how the models perform when confronted with unseen data, both extrapolation (*in-dis-test*) and different boundary condition (*out-dis-test*). For both test cases, all models perform significantly worse compared to the training phase. Nevertheless, FINN produces the best predictions with the lowest MSE, surpassing the other models by several orders of magnitude. More importantly, FINN is the only model with a consistently low prediction error with the same order of magnitude for all *train*, *in-dis-test*, and *out-dis-test* data. This model performance comparison is also visualized in Figure 4 for better clarity.

We also plot the predictions of the considered models for better visualization and understanding. For conciseness, we only show the plots for the dissolved concentration data generated with the Langmuir sorption isotherm due to similarities of the other plots. Figures 2 and 3 show the best prediction of each model, that produces the lowest MSE among the 10 randomly initialized trainings, when predicting the *in-dis-test* and the *out-dis-test* data, respectively. Figure 2 shows that most models, except for TCN and CNN-NODE, have at least one trained version that produces acceptable results even after extrapolation to a significantly longer time span (T = 10,000 days) compared to the one covered during training (T = 2,500 days). More interestingly and importantly, Figure 3 shows that when the boundary condition is changed to $c_{sol} = 0.7$, all models still tend to overfit to the boundary condition value used during training, that is, $c_{sol} = 1.0$, demonstrating the tendency of all models to overestimate the dissolved concentration close to $x = 0$. One distinguishing feature of FINN is its ability to properly treat different values of numerical boundary conditions, and thus, FINN is the only model that does not suffer from this overfitting issue, as also shown in Figure 2. Plots for data generated with other sorption isotherms and for the total concentration are presented in Supporting Information S1.

To assess the robustness of the considered ML models, Figures 5 and 6 show the prediction averaged over the 10 different training initializations for the *in-dis-test* and *out-dis-test* data, respectively. Moreover, Figures 5 and 6 are also equipped with 95% confidence intervals, to show the consistency of each model. These intervals are approximated with the t-distribution (Oliphant, 2006). Figure 5 shows that, even though each model has at least one good result, the other training result can still produce incorrect predictions. To be more specific, the average predictions by TCN, ConvLSTM, and CNN-NODE do not fit the data. Additionally, we observe that most of the models produce highly inconsistent predictions with wide confidence intervals. Figure 6 (for *out-dis-test*) shows worse consistency of all models, evidenced by the wider confidence intervals. Additionally, all existing ML models still overfit to the boundary condition value used in the *train* data, as discussed earlier. FINN, in contrast, produces very consistent predictions, making the confidence interval hardly visible in Figure 5. Furthermore, FINN shows excellent consistency and adjustment to the new boundary condition value in Figure 6. The performance comparison between FINN and CNN-NODE emphasizes the relevance of FINN's modularized structure. Apparently, NODE alone does not guarantee accurate function approximations, which is reflected in the larger training errors of CNN-NODE compared to FINN, as well as in the test errors—consistently over all experiments.

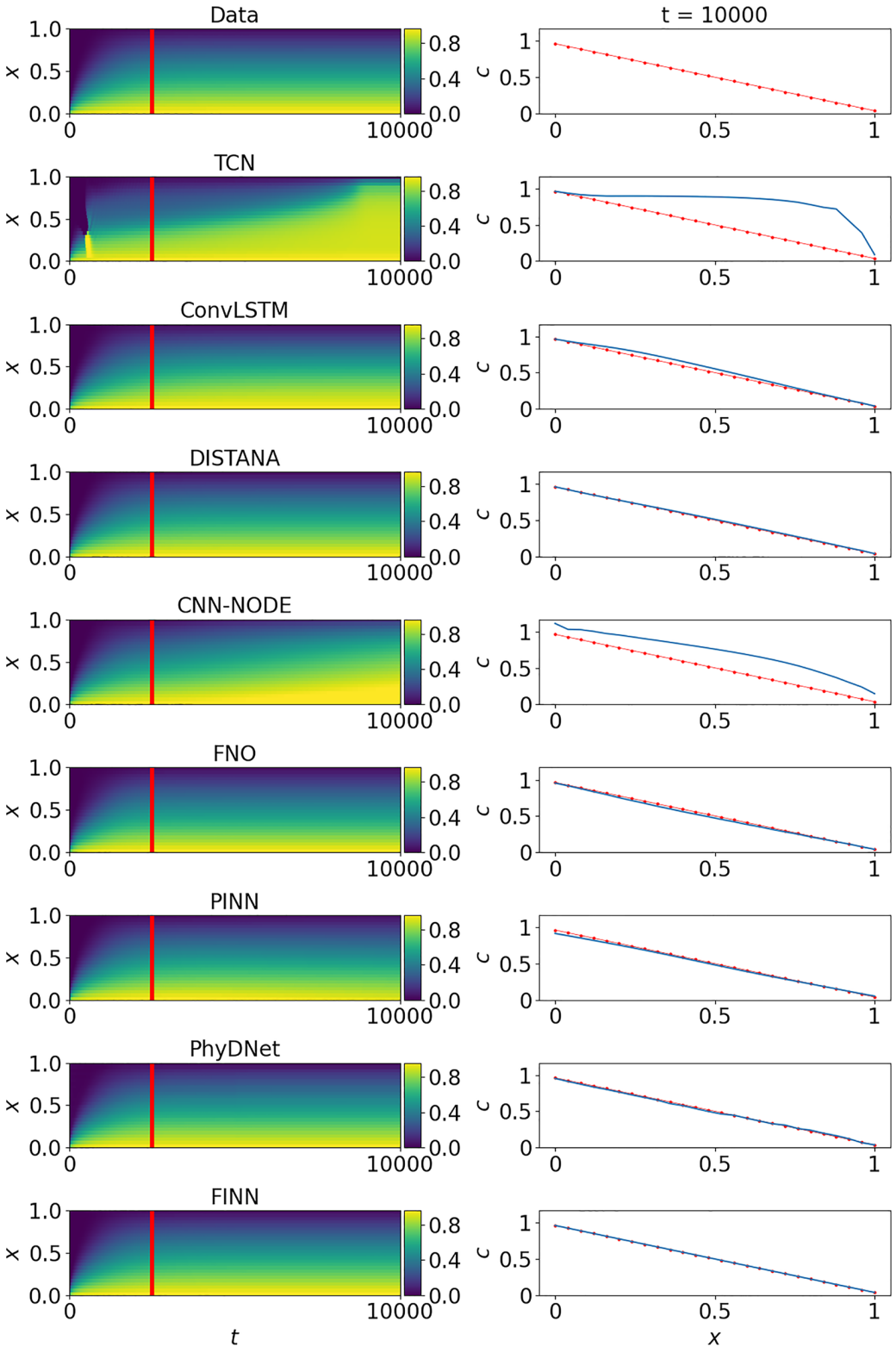


Figure 2. Plots of the dissolved concentration data generated with the Langmuir isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10{,}000$.

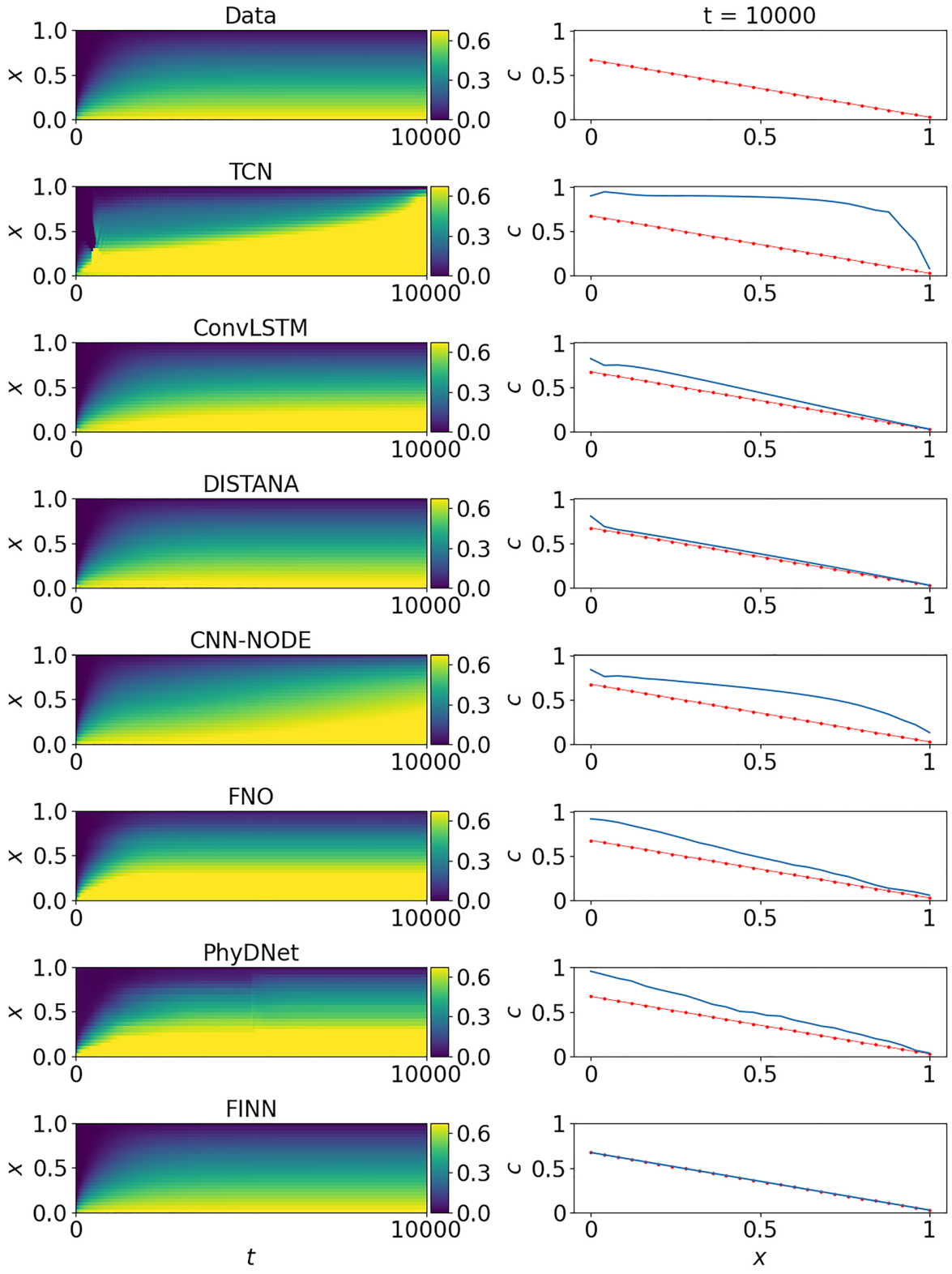


Figure 3. Plots of the dissolved concentration data generated with the Langmuir isotherm (red) (as well as a different boundary condition) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10{,}000$.

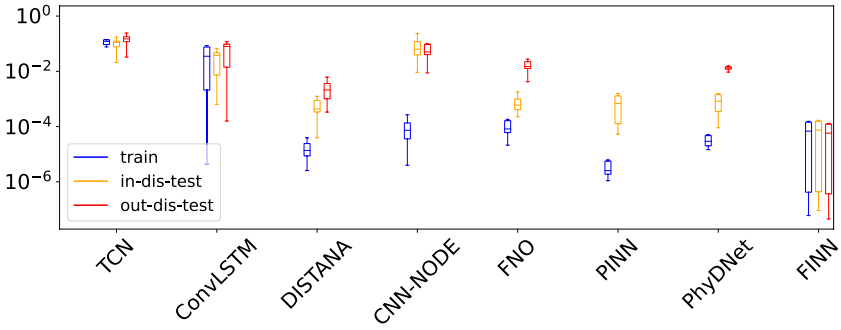


Figure 4. Average mean squared error (MSE) comparison of different Machine learning models with the error bars denoting the MSE standard deviation. The MSEs are calculated on the *train* (blue), *in-dis-test* (yellow), and *out-dis-test* (red) data set generated using the Langmuir isotherm.

We show with the example that a structurized method to design the model using the FVM discretization as a basis is extremely beneficial.

Focusing on the physics-aware ML modeling concepts, we could state that PINN and PhyDNet lie on different extremes. Even though PINN has a data-driven feature, it is more dominated by the physics-informed feature. It also can be formulated as an inverse problem, for example, to estimate unknown parameters such as the diffusion coefficient, but it still requires the modeler to know the complete form of the PDE to be solved. As a consequence, the retardation factor function also have to be known in advance to train the model. In contrast to PINN, PhyDNet puts more emphasis on the data-driven part, shown by the high number of parameters in the ConvLSTM branch compared to the physics-aware branch. Therefore, PhyDNet has more freedom in learning, but can suffer from overfitting issues. This is shown by the fact that PhyDNet achieves a very low prediction error during training, but its result significantly deteriorates when predicting *in-dis-test* and *out-dis-test* data. The introduced FINN concept lies somewhere in the middle of these extremes, compromising between the freedom of learning and the rigidity of (assumed) physical knowledge. As a result, FINN outperforms the other models, especially on the *out-dis-test* data, which is considered a particularly challenging task for ML models. Finally, while FNO can—in contrast to PINN—still be applied to different initial and boundary conditions, it suffers from a noticeable performance drop when applied to the new boundary condition. This is not surprising, as the explicit function learned during training (mapping continuous space-time coordinates to c) does not hold in the *out-dis-test* scenario anymore. This drawback of FNO could potentially be alleviated by embedding a more robust physics-informed prior in FNO, which results in the PINO model (Z. Li et al., 2021).

4.1.3. Learning the PDE Constituents With FINN

The most important feature of FINN is its ability to interpretably learn the building blocks of the sought PDE. In our example, the numerical stencil, the diffusion coefficient and the retardation factor function are learned during the training process. The learned numerical stencils and the learned diffusion coefficient D for the linear,

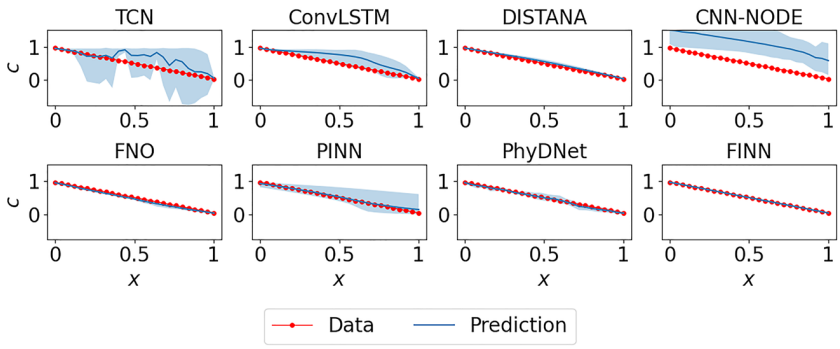


Figure 5. Prediction mean over 10 different trained models (with 95% confidence interval) of the dissolved concentration generated using the Langmuir isotherm at t = 10,000 for the *in-dis-test* data set.

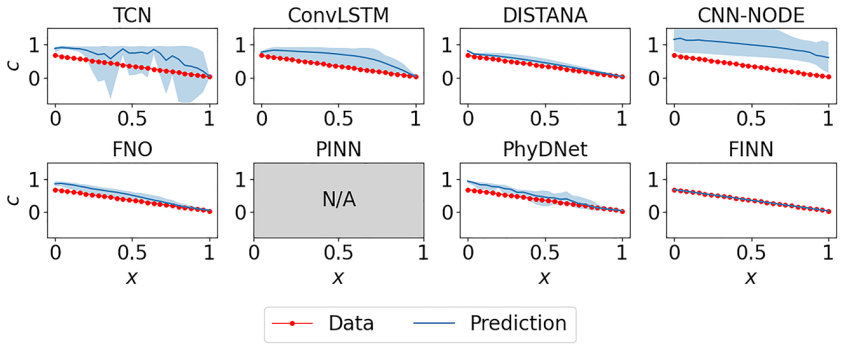


Figure 6. Prediction mean over 10 different trained models (with 95% confidence interval) of the dissolved concentration generated using the Langmuir isotherm at $t = 10{,}000$ for the *out-dis-test* data set. Note that physics-informed neural network (PINN) is not valid for predictions on a different boundary condition.

Freundlich, and Langmuir sorption isotherm data are shown in the second and third column of Table 4. All the learned numerical stencils are symmetrical, meaning that the prediction is mass-conservative and clearly diffusive.

Assuming that the ideal numerical stencils should be -1 and 1, we normalize the learned diffusion coefficient (by multiplying it with the learned numerical stencils) and compare it to the real diffusion coefficient value ($D = 5.0 \times 10^{-4}$ m^2/day) to evaluate the prediction error. The normalized diffusion coefficient for the linear, Freundlich, and Langmuir data are shown in the last column of Table 4. In terms of relative error to the real D value, these amount to 2.6%, 4.4%, and 10.8% error for the linear, Freundlich, and Langmuir isotherm, respectively. The prediction of the diffusion coefficient values has the highest error and variance for the Langmuir data, and the lowest error and variance for the linear data. As a consequence, FINN predicted the linear data with the highest accuracy, and Langmuir data with the lowest accuracy (see Table 2). Nevertheless, FINN's prediction is still highly consistent, as shown by the low MSE in Table 2 and the almost invisible confidence interval in Figures 5 and 6.

Furthermore, a very strong advantage of FINN is its ability to successfully learn closure/constitutional relationships, which are often unknown when modeling a system. In our diffusion-sorption example, the major source of uncertainty is the retardation factor $R(c)$, which can be defined with various empirical functions. Figure 7 shows the retardation factor learned by FINN, when trained on data generated with the three aforementioned sorption isotherms. FINN is able to learn the retardation factor through its module φ_D. The learned retardation factor also captures the linearity of the linear sorption isotherm, shown by the straight red line on the left plot in Figure 7. The retardation factors from the Freundlich and Langmuir isotherms are also captured well, even with less accuracy compared to the linear isotherm. This also contributes to the slightly higher prediction error for the Freundlich and Langmuir data compared to the linear data. Nevertheless, FINN is still able to distinguish between these different isotherms very well. Higher accuracy would need more informative data, especially at larger values of c.

4.2. Learning Using an Experimental Data Set

Synthetic data sets provide good insights into FINN's performance in a controlled experiment, where the data set is clean and abundant. However, real-world data is often only sparsely available due to costs or restrictions

Table 4
Learned Partial Differential Equation Constituents by Using Finite Volume Neural Network

Sorption isotherm	Numerical stencil	D (m^2/day)	Normalized D (m^2/day)
Linear	-1.06 ± 0.02 and 1.06 ± 0.02	$4.59 \pm 0.19 \times 10^{-4}$	$4.87 \pm 0.19 \times 10^{-4}$
Freundlich	-0.99 ± 0.04 and 0.99 ± 0.04	$4.83 \pm 0.19 \times 10^{-4}$	$4.78 \pm 0.34 \times 10^{-4}$
Langmuir	-1.08 ± 0.06 and 1.08 ± 0.06	$4.13 \pm 0.19 \times 10^{-4}$	$4.46 \pm 0.74 \times 10^{-4}$

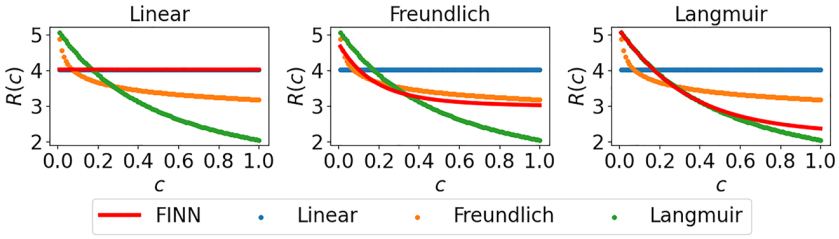


Figure 7. Learned retardation factor of finite volume neural network (FINN) for the linear (left), Freundlich (middle), and Langmuir (right) sorption isotherm, compared with the retardation factor generated with the three isotherms.

of equipments, the amount of time required to obtain useful data, or the difficulty in direct measurement of the system's internal states.

In the experimental setup described in Section 2.1, the dissolved TCE concentration distribution inside the clay sample is unobservable throughout the experiment. The only means of measuring the dissolved TCE concentration is through the water flushing below the lower end of the clay cylinder, as of now called a breakthrough curve. Furthermore, the total TCE concentration can only be measured at the end of the experiment, by cutting the clay specimen into slices to enable direct (but destructive) measurement of the total TCE concentration. Hence, the dissolved concentration data is available as a breakthrough curve, which has only a single data point at each time step; and the spatial distribution of the total concentration data is available only at the final time step, and at coarse spatial resolution. Additionally, the observation data obtained from the experiment is very sparse and also noisy. All these challenges associated with the use of real-world data prompt the implementation of UQ methods on FINN to provide honest and reliable predictions that would be useful for aiding critical decision making processes or hypothesis testing.

For this real-world application, three core samples are retrieved from the same geographical area, namely core samples #1, #2, and #2B (Nowak & Guthke, 2016). Consequently, similar soil parameters can be assumed for all three samples, which are summarized in Table 5. The breakthrough curve of core sample #2 is the least noisy, and hence is chosen as the training data; whereas the breakthrough curve of core #1 is chosen to test the trained model. Additional test is also performed with the data from core sample #2B. However, core #2B is significantly longer than the other samples and the bottom of the setup is closed (no flushing). By the end of the experiment, no measurable TCE has yet arrived at the bottom end of the sample. Numerically, the experiment is modeled with the setup as described in Section 2.1, with the initial condition written in Equation 5 and the boundary conditions written in Equations 3 and 4 for both core samples #1 and #2. For core sample #2B, because it is closed on the bottom, a no-flow Neumann boundary condition is used instead:

$$\left.\frac{\partial c}{\partial x}\right|_{x=L} = 0 \quad \forall t : 0 \le t \le T. \tag{32}$$

Table 5
Parameter Values of Various Clay Core Samples for the Laboratory Experiment

Parameter	Unit	Core #1	Core #2	Core #2B
Soil parameters				
D	m^2/day	2.00×10^{-5}	2.00×10^{-5}	2.78×10^{-5}
ϕ	–	0.288	0.288	0.288
ρ_s	kg/m^3	1,957	1,957	1,957
Simulation domain				
L	m	0.0254	0.02604	0.105
r	m	0.02375	0.02375	N/A
T	days	38.81	39.82	48.88
Q	m^3/day	1.01×10^{-4}	1.04×10^{-4}	N/A
c_{sol}	kg/m^3	1.4	1.6	1.4

The breakthrough curve of core #2 used as training data only serve for model training using the Cauchy boundary condition described in Equation 4. As a consequence, no other benchmark models can be used, since all of them, except for PINN, have no means of properly implementing numerical boundary conditions other than Dirichlet or periodic. PINN also cannot be applied in this example, because the test data set from core sample #1 and #2B have different boundary conditions, and therefore the PINN model trained on core sample #2 no longer holds for the other samples. Moreover, one of the most interesting goals of this experiment is to learn the retardation factor, whereas all the considered ML models have no capability to do so explicitly. Therefore, we assess the performance of FINN using a comparison to the PDE-based physical model—the same as used to generate synthetic data in Section 4.1—calibrated to the experimental data as a benchmark. The best fit of the physical model is found with the retardation factor modeled using the Freundlich sorption isotherm, with $K_f = 5.20 \times 10^{-4}\,(m^3/kg)^{n_f}$ and $n_f = 0.35$. Note that this Freundlich exponent is considered low, and can be attributed

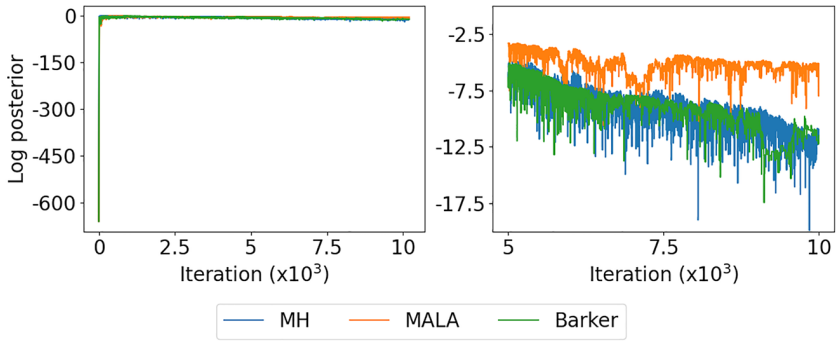


Figure 8. Trace plot of the log posterior starting with random initial values (left) and the zoomed-in plot after 5,000 iterations (right) for the Metropolis Hastings (MH, blue), Metropolis-adjusted Langevin algorithm (MALA, orange), and Barker (green) MCMC methods.

to the scarcity of the training data. This also shows that even fitting a well-defined physical model to limited data could lead to inaccuracies.

For this application, FINN is implemented with the use of the module φ_D. Here, the module φ_D is defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes c as an input and outputs the retardation factor $R(c)$. The diffusion coefficient is assumed to be known and measurable for all the core samples (Nowak & Guthke, 2016), and therefore is not learned by FINN. As in the synthetic case, FINN is trained with the objective to minimize the deviation between the model predictions of c as a breakthrough curve and the total concentration c_t at the end of the experiment. Also, in contrast to the synthetic data scenario, FINN is now trained using the Bayes-by-backprop method as outlined in Section 2.4 using Equation 19 as the loss function formulation. Even though the Bayes-by-backprop method manages to provide reasonable UQ of FINN's prediction, it fails to learn the standard deviation parameter σ sufficiently (i.e., the learned σ values do not differ much from the initial values).

Due to the limitations of the Bayes-by-backprop method, we alternatively use three different MCMC methods, namely the random walk MH, MALA, and Barker. Starting with random initial values of FINN's parameters, samplings are performed with these three methods for 102,000 iterations, with the 2,000 initial iterations discarded as the burn-in period, resulting in 100,000 effective iterations. This number is chosen as the upper limit, to investigate which MCMC method provides decent convergence and offers the most efficient sampling under acceptable computational time.

Out of the 100,000 iterations, we thin out the samples by saving only every tenth iteration, resulting in a total of 10,000 samples. The step size h is chosen so that the acceptance rate amounts to approximately 23% (Reuschen et al., 2021). This corresponds to $h = 10^{-2}$ for MH, $h = 7 \times 10^{-6}$ for MALA, and $h = 4 \times 10^{-3}$ for Barker. As shown by the trace plot (the left plot in Figure 8), all methods improve the log posterior substantially after only a few iterations. However, looking at the zoomed-in plot (the right plot in Figure 8), none of the used MCMC methods converge well to an equilibrium distribution, as evidenced by the downward trend of the log posterior even until the last iteration. Even though MALA has better convergence compared to the other methods, the acceptance rate quickly deteriorates. This can be attributed to the fact that MALA is less robust to high step size (Livingstone and Zanella (2019), although the step size used for MALA is already very low in this case).

To improve the performance of the MCMC chains, we start the sampling with optimized parameter values of FINN. To obtain this, FINN is first trained deterministically as in the synthetic data scenario, and the parameter values of the trained FINN model are used as the starting point of the MCMC chain. Samplings are then performed for 100,000 iterations, without the burn-in period (since we start with optimized values), again thinning by a factor of 10, resulting in a total of 10,000 samples. The corresponding step sizes are $h = 10^{-2}$ for MH, $h = 3 \times 10^{-5}$ for MALA, and $h = 5 \times 10^{-3}$ for Barker.

As shown by the log posterior trace plot (the left plot in Figure 9), there is a downward trend in the log posterior of the samples. This is fundamentally caused by using an excellent starting point with minimized error (statistically too good to be a representative sample), resulting in less good (but as of then statistically valid) subsequent

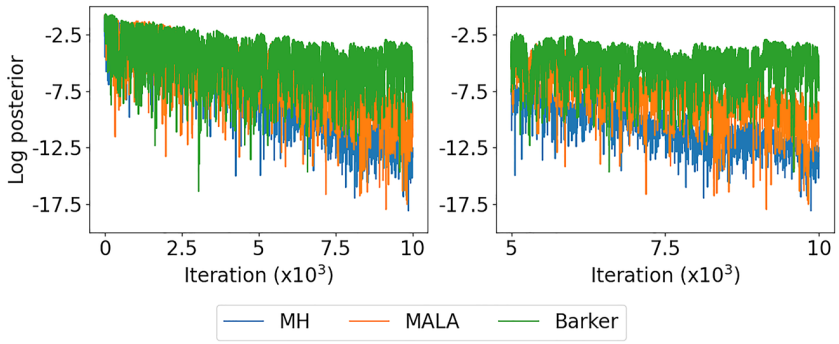


Figure 9. Trace plot of the log posterior starting with optimized initial values (left) and the zoomed-in plot after 5,000 iterations (right) for the Metropolis Hastings (MH, blue), Metropolis-adjusted Langevin algorithm (MALA, orange), and Barker (green) Markov chain Monte Carlo methods.

samples. When zooming in after 5,000 iterations, we observe that the performance of both the gradient-based MCMC methods, namely MALA and Barker, is better than that of MH. It is also worth to note that the step size tuning for both MALA and Barker method is not straightforward due to the influence from the loss gradient on the jump distribution, and therefore, could potentially lead to a heavier computational burden.

Among these two gradient-based MCMC methods, Barker shows the best behavior with the highest and most stable log posterior values, indicating proper sampling from the desired posterior. On the other hand, the log posterior value of the samples obtained using MALA is still slightly decreasing. This result shows that Barker scales well for higher dimensionality, especially when the chain is properly initialized. All in all, only the optimized Barker MCMC finishes its burn-in properly and reaches an equilibrium distribution, that is the desired posterior, within the 100,000 iterations window.

Another way to quantify the predictive performance/sharpness of the MCMC methods is to plot the reliability curve as defined in Jospin et al. (2022), which is calculated using the cumulative distribution function across all samples, compared to its observed probability (i.e., ordered against the actual data). The reliability plot enables evaluation of the model's predictive performance. The model is described as underconfident if the reliability curve lies above the baseline, and overconfident otherwise. As shown in Figure 10, all the methods with random initialization (left plot) lie further from the ideal condition compared to the methods with optimized starting point (right plot). Among all the methods, Barker MCMC with optimized initialization lies the closest to the ideal condition, confirming further that the samples generated by the optimized Barker MCMC provides the best predictive performance. It is also interesting to note that the models are overconfident for lower extremes and underconfident for higher extremes. One possible explanation is that the data error is not Gaussian.

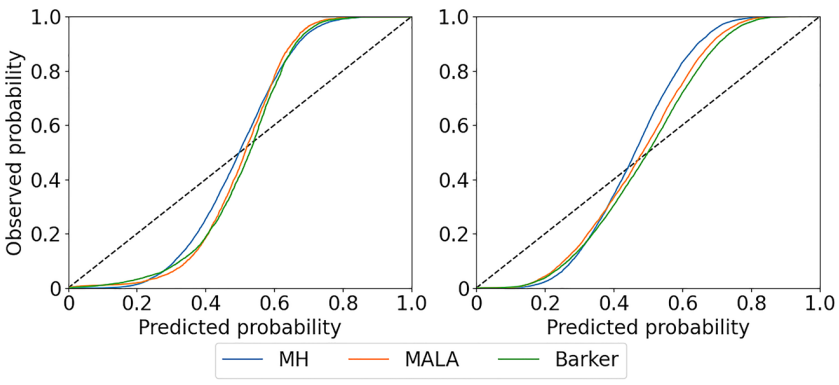


Figure 10. Reliability curves of the Metropolis Hastings (MH, blue), Metropolis-adjusted Langevin algorithm (MALA, orange), and Barker (green) Markov chain Monte Carlo methods initialized randomly (left) and with optimized values (right). The baseline for the ideal condition is shown by the black dashed line.

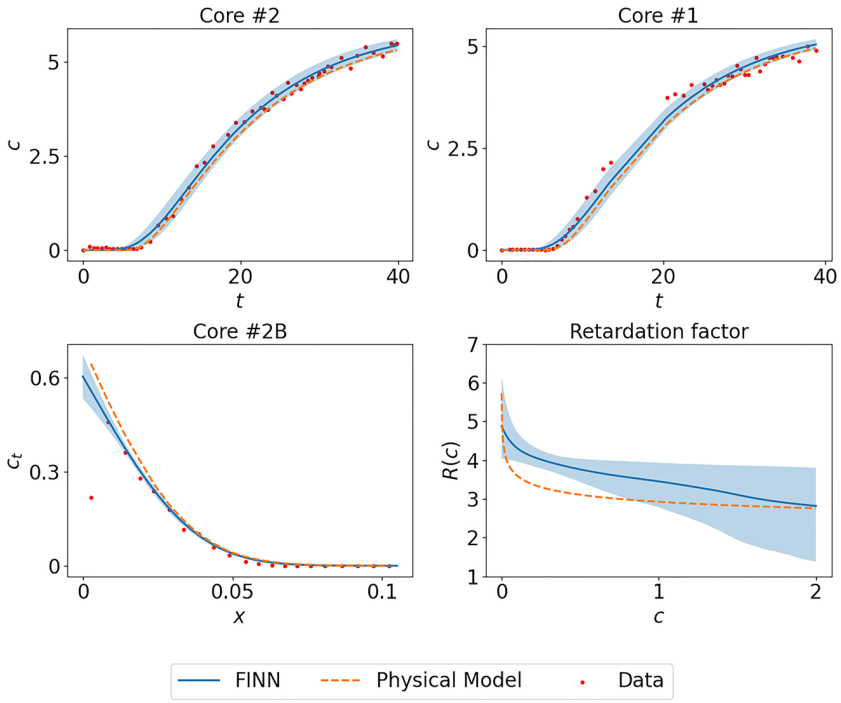


Figure 11. Breakthrough curve average prediction of finite volume neural network (FINN, blue line) and its 95% confidence interval (blue shade) during training on core sample #2 (top left), during testing on core sample #1 (top right) and total concentration profile of core sample #2B (bottom left). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The learned retardation factor $R(c)$ is shown in the bottom right plot.

The predictions obtained using the optimized Barker MCMC are shown in Figure 11. The MCMC method augment FINN's prediction with a confidence interval, which captures most of the noisy observation data inside, showing sufficient UQ. Quantitatively, FINN achieves lower training error on core sample #2 data with MSE $= 5.43 \times 10^{-4}$, compared to the physical model with MSE $= 1.06 \times 10^{-3}$. During testing with data from core sample #1, FINN also outperforms the physical model with MSE $= 1.41 \times 10^{-3}$ compared to MSE $= 2.50 \times 10^{-3}$, because the calibrated physical model underestimates the TCE breakthrough curve. When tested against data from core sample #2B, which has a different type of numerical boundary condition implemented, FINN again achieves lower prediction error with MSE $= 1.16 \times 10^{-3}$ compared to the calibrated physical model that overestimates the TCE concentration with MSE $= 2.73 \times 10^{-3}$. Because there is no breakthrough curve data available for this specific sample, we compare the prediction against the total concentration profile $c(x, t = T)$ at the end of the experiment. The differences between FINN and the physical model prediction are not very clear in the breakthrough curve plots because both predictions have only small variance at the end of the sample. However, the error of the physical model prediction becomes more apparent in the total concentration profile (Figure 11, bottom left).

Moreover, we also plot FINN's learned retardation factor in comparison to the calibrated Freundlich retardation factor, which shows that the best-fitting available sorption isotherm model fails to capture the retardation factor shape as learned by FINN, possibly leading to the higher prediction error of the calibrated physical model in both cases of training and testing. Overall, FINN outperforms the calibrated physical model by learning the retardation factor better than the parametric sorption isotherm model using only the breakthrough curve of core sample #2 (i.e., only 55 data points) and successfully applies it to the other samples with relatively high accuracy.

As a side note, this real-world application example adopted in this work was performed in a small-scale laboratory experiment. With the corresponding scale, homogeneity could be assumed for the modeled soil parameters. For a larger-scale application (i.e., field-scale), the assumption might no longer hold, and thus heterogeneity would have to be taken into account. To account for heterogeneity, FINN has to either adopt a geostatistical approach to model the heterogeneous distribution of the parameter (e.g., the diffusion coefficient), or a graph representation.

Finally, even though we only showed the application of FINN to a subsurface contaminant transport problem, it is also applicable to a range of other problems or equations, such as the 2D Burgers' the diffusion-reaction equation, and the Allen-Cahn equation. For further details, we refer the interested readers to our ML-focused paper (Karlbauer et al., 2022). FINN lays the groundwork for further development of hybrid modeling frameworks in this area, and hopefully can be used for an even wider range of problems in the future, such as weather and climate simulation or investigation of improved constitutive relations in multiphase flow.

5. Summary and Conclusion

In this work, we applied FINN, a hybrid modeling framework that induces physical inductive biases into an ANN learning paradigm. FINN is based on the numerical structure of the FVM for solving PDEs, as well as on conditions such as monotonicity and non-negativity of functions or parameters to constrain the model training. FINN learns numerical stencils, unknown constitutional/closure relationships, and/or parameters to predict variables of interest in spatiotemporal physical systems. Using a well-controlled subsurface contaminant transport benchmark, we showed that FINN is beneficial in comparison to pure ML models as well as physics-motivated and physics-aware ML models for several reasons.

First, FINN demonstrates superior generalization when tested against extrapolated data and data generated with different boundary condition. The other ML models participating in our comparison have the tendency to overfit, while PINN is not even applicable to the same system with a different boundary condition.

Second, FINN allows proper treatment of different boundary condition types, whereas other models are only applicable to boundary condition types with constant values such as Dirichlet or periodic ones, because of the convolutional structure adopted in most of the other models. As a result, only FINN can be trained on data under a Cauchy boundary condition in the form of a diffusive breakthrough curve in the real-world experimental data example.

Third, FINN can be trained with a sparse data set without compromising its learning ability and its prediction accuracy. It was shown in the real-world data example that FINN was trained with only 55 data points, yet it generalized well to other unseen samples, even one with a different boundary condition type. This also implies that FINN is applicable to real-world data that is noisy and sparse. Hence, FINN offers a data-driven modeling approach that goes beyond a surrogate modeling tool.

Fourth, FINN provides flexibility in choosing between different UQ methods, especially due to its comparatively low number of parameters. FINN can be paired not only with the variational inference type of UQ (i.e., Bayes-by-backprop), but also with MCMC methods. Additionally, the widely available automatic differentiation tools in various ML libraries enable the use of gradient information in both MALA and Barker MCMC, leading to better performance compared to a random walk MH. Furthermore, these automatic differentiation tools also promote finding an optimal starting point for the MCMC chain, by first training FINN deterministically.

Fifth, and probably most importantly, FINN's structure provides a high degree of model explainability. Through its structure, FINN can explicitly learn unknown constitutive/closure relationships or parameters, which are usually the main source of uncertainty in physical systems modeling. The particular example shown in this work is the unknown retardation factor as a function of concentration. Because the available sorption isotherm models describe the retardation factor function using few parameters, they are not flexible enough to be calibrated to learn the “true” shape of the function. FINN, on the other hand, has the full flexibility to learn it.

To re-emphasize, in this work we did not intend to develop FINN as a faster and more efficient surrogate model in place of known equations and their numerical solvers. The only comparison between FINN and a traditional numerical simulation model was presented on the real-world problem example. The purpose of this comparison was to show that, when calibrating the physical model, we are faced with discrete choices of models that lead to difficulties in capturing the “true” functional relationship. FINN, on the other hand, alleviates this problem by providing a flexible way of learning the unknown relationship, as shown in Section 4.2.

Despite the promising benefits of FINN, we realized that there is still a lot of room for improvement. For instance, the computation time of FINN is still not optimized, because the implementation is highly dependent on the available NODE package. It will not be as fast as PINN, because PINN models the system as an explicit function

of x and t, allowing to parallelize the computation. The recurrent structure in FINN and other models, such as ConvLSTM, DISTANA, and PhyDNet, on the other hand, prohibit full parallelization.

Additionally, FINN offers room for possible extension of the underlying physical theory behind the framework. For example, it could be enabled to account for non-Fickian diffusion or dispersion, where the diffusion coefficient can be a highly complex function of many variables, or for other neglected processes that may contribute significantly to the modeled system, such as chemical transformation, evaporation, or advection. In these cases, FINN's structure would need to be modified to allow for a more complicated representation of the unknown functional relationships. This could be achieved, for example, not only by including more independent variables as input to the different neural network modules, but also spatio-temporal dynamics of more dependent variables to be approximated by FINN (e.g., including more non-linearly coupled PDEs).

To upscale FINN to a field size domain, spatial heterogeneity also has to be taken into account, due to the high possibility of higher variations of the geophysical properties across a larger area. It is possible to model a heterogeneous system either by modeling the diffusion coefficient as a function of space, complemented by geostatistical methods, or by modifying FINN's neural network modules to be connected via a graph-like structure. This will allow for more freedom in the modeled interactions and the message passing between neighboring spatial control volumes as well as in implementations on unstructured grids.

UQ of ML models is still a broad and open research area, due to the complicated nature of ML models, and ANNs in particular. More rigorous analysis of UQ on FINN can further improve the predictive performance and foster the interpretability of the model itself. Furthermore, it would also be beneficial to reach a fully accurate total UQ over inferred scientific hypotheses. This could include not only the uncertainty from parameters, closures, and stencils covered in this work, but also over entirely different conceptualizations of a system. The latter could result in different structural set-ups of FINN. Additional research is necessary to pursue these challenges.

Data Availability Statement

All codes used for generating the *train*, *in-dis-test*, and *out-dis-test* data and reproduce all the results in this paper are preserved at https://doi.org/10.5281/zenodo.7260671, available via the MIT license and developed openly at https://github.com/CognitiveModeling/finn (Praditia et al., 2022).

Acknowledgments
This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC 2075–390740016 as well as EXC 2064–390727645. The authors acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). Moreover, the authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer. Open Access funding enabled and organized by Projekt DEAL.

References

Al-Ghouti, M. A., & Da'ana, D. A. (2020). Guidelines for the use and interpretation of adsorption isotherm models: A review. *Journal of Hazardous Materials*, *393*, 122383. https://doi.org/10.1016/j.jhazmat.2020.122383

Allen, S. M., & Cahn, J. W. (1979). A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metallurgica*, *27*(6), 1085–1095. https://doi.org/10.1016/0001-6160(79)90196-2

Allen-King, R. M., Groenevelt, H., James Warren, C., & Mackay, D. M. (1996). Non-linear chlorinated-solvent sorption in four aquitards. *Journal of Contaminant Hydrology*, *22*(3), 203–221. https://doi.org/10.1016/0169-7722(95)00089-5

Almqvist, O. (2019). A comparative study between algorithms for time series forecasting on customer prediction: An investigation into the performance of ARIMA, RNN, LSTM, TCN and HMM. Retrieved from http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1321224

Arfken, G., Weber, H., & Harris, F. (2013). *Mathematical methods for physicists: A comprehensive guide*. Elsevier Science. Retrieved from https://books.google.de/books?id=qLFo_Z-PoGIC

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*. https://doi.org/10.48550/arXiv.1803.01271

Bardenet, R., Doucet, A., & Holmes, C. C. (2017). On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, *18*(47), 1–43. Retrieved from https://www.jmlr.org/papers/volume18/15-205/15-205.pdf

Bar-Sinai, Y., Hoyer, S., Hickey, J., & Brenner, M. (2019). Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, *116*(31), 15344–15349. https://doi.org/10.1073/pnas.1814058116

Battaglia, P. W., Hamrick, J., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*. https://doi.org/10.48550/arXiv.1806.01261

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., & Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*. https://doi.org/10.48550/arXiv.1612.00222

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877. https://doi.org/10.1080/01621459.2017.1285773

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning – Volume 37* (pp. 1613–1622). Retrieved from http://proceedings.mlr.press/v37/blundell15.pdf

Brandstetter, J., Worrall, D. E., & Welling, M. (2022). Message passing neural PDE solvers. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=vSix3HPYKSU

Brown, G. H., Brooks, M. C., Wood, A. L., Annable, M. D., & Huang, J. (2012). Aquitard contaminant storage and flux resulting from dense nonaqueous phase liquid source zone dissolution and remediation. *Water Resources Research*, *48*(6), W06531. https://doi.org/10.1029/2011WR011141

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*. https://doi.org/10.48550/arXiv.2005.14165

Butcher, J. (2008). *Numerical methods for ordinary differential equations*. Wiley.

Chen, R., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*. https://doi.org/10.48550/arXiv.1806.07366

Cheng, A. H.-D., & Cheng, D. T. (2005). Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements*, *29*(3), 268–302. https://doi.org/10.1016/j.enganabound.2004.12.001

Chib, S., & Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, *49*(4), 327–335. Retrieved from http://www.jstor.org/stable/2684568

Courant, R., Friedrichs, K., & Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, *11*(2), 215–234. https://doi.org/10.1147/rd.112.0215

De Bézenac, E., Pajot, A., & Gallinari, P. (2019). Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, *2019*(12), 124009. https://doi.org/10.1088/1742-5468/ab3195

Dwivedi, R., Chen, Y., Wainwright, M. J., & Yu, B. (2019). Log-concave sampling: Metropolis-Hastings algorithms are fast. *Journal of Machine Learning Research*, *20*(183), 1–42. Retrieved from http://jmlr.org/papers/v20/19-306.html

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., et al. (2021). Skillful twelve hour precipitation forecasts using large context neural networks. *arXiv preprint arXiv:2111.07470*. https://doi.org/10.48550/arXiv.2111.07470

Fetter, C. W. (1999). *Contaminant hydrogeology*. Prentice Hall.

Fornberg, B. (1988). Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, *51*(184), 699–706. https://doi.org/10.1090/s0025-5718-1988-0935077-0

Ghosh, T., Bringedal, C., Helmig, R., & Sekhar, G. R. (2020). Upscaled equations for two-phase flow in highly heterogeneous porous media: Varying permeability and porosity. *Advances in Water Resources*, *145*, 103716. https://doi.org/10.1016/j.advwatres.2020.103716

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning – Volume 70* (pp. 1263–1272). Retrieved from https://dl.acm.org/doi/10.5555/3305381.3305512

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT Press.

Guen, V., & Thome, N. (2020). Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and pattern Recognition* (pp. 11474–11484). Retreieved from https://openaccess.thecvf.com/content_CVPR_2020/papers/Le_Guen_Disentangling_Physical_Dynamics_From_Unknown_Factors_for_Unsupervised_Video_Prediction_CVPR_2020_paper.pdf

Hayduk, W., & Laudie, H. (1974). Prediction of diffusion coefficients for nonelectrolytes in dilute aqueous solutions. *AIChE Journal*, *20*(3), 611–615. https://doi.org/10.1002/aic.690200329

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7780459

He, Q., Barajas-Solano, D., Tartakovsky, G., & Tartakovsky, A. M. (2020). Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, *141*, 103610. https://doi.org/10.1016/j.advwatres.2020.103610

Hendry, M. J., Ranville, J. R., Boldt-Leppin, B. E. J., & Wassenaar, L. I. (2003). Geochemical and transport properties of dissolved organic carbon in a clay-rich aquitard. *Water Resources Research*, *39*(7), 1194. https://doi.org/10.1029/2002WR001943

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). *Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. A field guide to dynamical recurrent neural networks*. IEEE Press.

Höge, M., Guthke, A., & Nowak, W. (2019). The hydrologist's guide to Bayesian model selection, averaging and combination. *Journal of Hydrology*, *572*, 96–107. https://doi.org/10.1016/j.jhydrol.2019.01.072

Huang, J., & Goltz, M. N. (2015). Semianalytical solutions for transport in aquifer and fractured clay matrix system. *Water Resources Research*, *51*(9), 7218–7237. https://doi.org/10.1002/2014WR016073

IPCC. (2013). *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel On Climate Change*. In T. Stocker, D. Qin, G.-K. Plattner, M. Tignor, S. K. Allen, J. Boschung, et al. (Eds.). Cambridge University Press.

Isaacson, E., & Keller, H. (1994). *Analysis of numerical methods*. Dover Publications.

Jin, X., Cai, S., Li, H., & Karniadakis, G. (2021). NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, *426*, 109951. https://doi.org/10.1016/j.jcp.2020.109951

Johnson, G. R., Zhang, Z., & Brusseau, M. L. (2003). Characterizing and quantifying the impact of immiscible-liquid dissolution and nonlinear, rate-limited sorption/desorption on low-concentration elution tailing. *Water Resources Research*, *39*(5), 1120. https://doi.org/10.1029/2002WR001435

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, *17*(2), 29–48. https://doi.org/10.1109/MCI.2022.3155327

Joyce, J. M. (2011). Kullback-Leibler divergence. In M. Lovric (Ed.), *International encyclopedia of statistical science* (pp. 720–722). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_327

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A., Graves, A., & Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*. https://doi.org/10.48550/arXiv.1610.10099

Karlbauer, M., Menge, T., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., & Butz, M. (2020). Hidden latent state inference in a spatio-temporal generative model. *arXiv preprint arXiv:2009.09823*. https://doi.org/10.48550/arXiv.2009.09823

Karlbauer, M., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., & Butz, M. (2019). A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint arXiv:1912.11141*. https://doi.org/10.48550/arXiv.1912.11141

Karlbauer, M., Praditia, T., Otte, S., Oladyshkin, S., Nowak, W., & Butz, M. V. (2022). Composing partial differential equations with physics-aware neural networks. In *Proceedings of the 39th International Conference on Machine Learning*. Retrieved from https://proceedings.mlr.press/v162/karlbauer22a.html

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, *3*(6), 422–440. https://doi.org/10.1038/s42254-021-00314-5

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*. Retrieved from http://arxiv.org/abs/1412.6980

Klaasen, G., & Troy, W. (1984). Stationary wave solutions of a system of reaction-diffusion equations derived from the Fitzhugh–Nagumo equations. *SIAM Journal on Applied Mathematics*, *44*(1), 96–110. https://doi.org/10.1137/0144008
Koch, J., & Nowak, W. (2015). Predicting DNAPL mass discharge and contaminated site longevity probabilities: Conceptual model and high-resolution stochastic simulation. *Water Resources Research*, *51*(2), 806–831. https://doi.org/10.1002/2014WR015478
Koch, T., Weishaupt, K., Müller, J., Weigand, B., & Helmig, R. (2021). A (dual) network model for heat transfer in porous media. *Transport in Porous Media*, *140*(1), 107–141. https://doi.org/10.1007/s11242-021-01602-5
Kochenderfer, M., & Wheeler, T. (2019). *Algorithms for optimization*. MIT Press. Retrieved from https://books.google.de/books?id=uBSMDwAAQBAJ
Kochkov, D., Smith, J., Alieva, A., Wang, Q., Brenner, M., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, *118*(21), e2101784118. https://doi.org/10.1073/pnas.2101784118
Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Neural operator: Learning maps between function spaces. https://doi.org/10.48550/ARXIV.2108.08481
Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105. https://doi.org/10.1145/3065386
Kutta, W. (1901). Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Zeitschrift für angewandte Mathematik und Physik*, *46*, 435–453.
Lake, B. (2019). Compositional generalization through meta sequence-to-sequence learning. In *Advances in neural information processing systems 32* (pp. 9791–9801). Retrieved from https://proceedings.neurips.cc/paper/2019/file/f4d0e2e7fc057a58f7ca4a391f01940a-Paper.pdf
Lake, B., Ullman, T., Tenenbaum, J., & Gershman, S. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, e253. https://doi.org/10.1017/S0140525X16001837
Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision* (pp. 47–54). https://doi.org/10.1007/978-3-319-49409-8_7
Li, K., & Horne, R. N. (2006). Comparison of methods to calculate relative permeability from capillary pressure in consolidated water-wet porous media. *Water Resources Research*, *42*(6), W06405. https://doi.org/10.1029/2005WR004482
Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020a). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*. https://doi.org/10.48550/arXiv.2010.08895
Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020b). Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*. https://doi.org/10.48550/arXiv.2003.03485
Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., et al. (2021). Physics-informed neural operator for learning partial differential equations. https://doi.org/10.48550/ARXIV.2111.03794
Lienen, M., & Günnemann, S. (2022). Learning the dynamics of physical systems from sparse observations with finite element networks. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=HFmAukZ-k-2
Limousin, G., Gaudet, J.-P., Charlet, L., Szenknect, S., Barthès, V., & Krimissa, M. (2007). Sorption isotherms: A review on physical bases, modeling and measurement. *Applied Geochemistry*, *22*(2), 249–275. https://doi.org/10.1016/j.apgeochem.2006.09.010
Livingstone, S., & Zanella, G. (2019). The Barker proposal: Combining robustness and efficiency in gradient-based MCMC. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, *84*(2), 496. https://doi.org/10.48550/ARXIV.1908.11812
Logan, D. (1992). *A first course in the finite element method*. PWS-Kent Publishing Company. Retrieved from https://books.google.de/books?id=qCiRQgAACAAJ
Long, Z., Lu, Y., Ma, X., & Dong, B. (2018). PDE-Net: Learning PDES from data. In *International Conference on Machine Learning* (pp. 3208–3216). Retrieved from https://proceedings.mlr.press/v80/long18a.html
Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, *3*(3), 218–229. https://doi.org/10.1038/s42256-021-00302-5
Lu, Y., Zhong, A., Li, Q., & Dong, B. (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning* (Vol. 80, pp. 3276–3285). PMLR. Retrieved from https://proceedings.mlr.press/v80/lu18d.html
Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning – Volume 20* (pp. 1–7). Association for Computational Linguistics. https://doi.org/10.3115/1118853.1118871
Marchuk, G. (1974). *Numerical methods in weather prediction*. Elsevier.
Moghadasi, L., Guadagnini, A., Inzoli, F., & Bartosek, M. (2015). Interpretation of two-phase relative permeability curves through multiple formulations and model quality criteria. *Journal of Petroleum Science and Engineering*, *135*, 738–749. https://doi.org/10.1016/j.petrol.2015.10.027
Morton, K., & Mayers, D. (1994). *Numerical solution of partial differential equations: An introduction*. Cambridge University Press. Retrieved from https://books.google.de/books?id=h-s-nwEACAAJ
Moukalled, F., Mangani, L., & Darwish, M. (2016). *The finite volume method in computational fluid dynamics* (1st ed.). Springer. https://doi.org/10.1007/978-3-319-16874-6
National Toxicology Program, US Department of Health and Human Services. (2021). *15th report on carcinogens* (Tech. Rep.). National Toxicology Program, US Department of Health and Human Services.
Nowak, W., & Guthke, A. (2016). Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, *18*(11), 409. https://doi.org/10.3390/e18110409
Oliphant, T. E. (2006). *A Bayesian perspective on estimating mean, variance, and standard-deviation from data*. Brigham Young University.
Pankow, J., & Cherry, J. (1996). *Dense chlorinated solvents and other DNAPLs in groundwater: History, behavior, and remediation*. Waterloo Press. Retrieved from https://books.google.de/books?id=OiYfAQAAIAAJ
Parker, B. L., Cherry, J. A., & Chapman, S. W. (2004). Field study of TCE diffusion profiles below DNAPL to assess aquitard integrity. *Journal of Contaminant Hydrology*, *74*(1), 197–230. https://doi.org/10.1016/j.jconhyd.2004.02.011
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32, pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
Praditia, T., Karlbauer, M., Otte, S., Oladyshkin, S., Butz, M. V., & Nowak, W. (2022). Finite volume neural network [Software]. Zenodo. https://doi.org/10.5281/zenodo.7260671
Praditia, T., Walser, T., Oladyshkin, S., & Nowak, W. (2020). Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture. *Energies*, *13*(15), 3873. https://doi.org/10.3390/en13153873

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). 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. https://doi.org/10.1016/j.jcp.2018.10.045

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., & Thuerey, N. (2020). WeatherBench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, *12*(11), e2020MS002203. https://doi.org/10.1029/2020ms002203

Reuschen, S., Jobst, F., & Nowak, W. (2021). Efficient discretization-independent Bayesian inversion of high-dimensional multi-Gaussian priors using a hybrid MCMC. *Water Resources Research*, *57*(8), e2021WR030051. https://doi.org/10.1029/2021WR030051

Runge, C. (1895). Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, *46*(2), 167–178. https://doi.org/10.1007/bf01446807

Salehi, Y., & Giannacopoulos, D. (2021). PhysGNN: A physics-driven graph neural network based model for predicting soft tissue deformation in image-guided neurosurgery. *arXiv preprint arXiv:2109.04352*. https://doi.org/10.48550/arXiv.2109.04352

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. W. (2020). Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning* (pp. 8459–8468). Retrieved from https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html

Seo, S., Meng, C., & Liu, Y. (2019). Physics-aware difference graph networks for sparsely-observed dynamics. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=r1gelyrtwH

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*. https://doi.org/10.48550/arXiv.1506.04214

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, *33*, 7462–7473. Retrieved from https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html

Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., & Barajas-Solano, D. (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, *56*(5), e2019WR026731. https://doi.org/10.1029/2019WR026731

Thuerey, N., Holl, P., Mueller, M., Schnell, P., Trost, F., & Um, K. (2021). *Physics-based deep learning*. WWW. Retrieved from https://physicsbaseddeeplearning.org

Timms, W. A., Acworth, R. I., Crane, R. A., Arns, C. H., Arns, J.-Y., McGeeney, D. E., et al. (2018). The influence of syndepositional macropores on the hydraulic integrity of thick alluvial clay aquitards. *Water Resources Research*, *54*(4), 3122–3138. https://doi.org/10.1029/2017WR021681

Tran, G., & Ward, R. (2017). Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling and Simulation*, *15*(3), 1108–1129. https://doi.org/10.1137/16m1086637

Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, *237*, 37–72.

Wei, J., & Winter, M. (2017). Stable spike clusters for the one-dimensional Gierer–Meinhardt system. *European Journal of Applied Mathematics*, *28*(4), 576–635. https://doi.org/10.1017/S0956792516000450

Wilke, C. R., & Chang, P. (1955). Correlation of diffusion coefficients in dilute solutions. *AIChE Journal*, *1*(2), 264–270. https://doi.org/10.1002/aic.690010222

Winsberg, E. (2003). Simulated experiments: Methodology for a virtual world. *Philosophy of Science*, *70*(1), 105–125. https://doi.org/10.1086/367872

Wöhling, T., Schöniger, A., Gayler, S., & Nowak, W. (2015). Bayesian model averaging to explore the worth of data for soil-plant model selection and prediction. *Water Resources Research*, *51*(4), 2825–2846. https://doi.org/10.1002/2014WR016292

World Health Organization, Regional Office for Europe. (2000). 5.15 trichloroethylene. In *Air Quality Guidelines for Europe* (2nd ed., pp. 115–117). World Health Organization, Regional Office for Europe.

Xu, T., Reuschen, S., Nowak, W., & Hendricks Franssen, H.-J. (2020). Preconditioned Crank-Nicolson Markov chain Monte Carlo coupled with parallel tempering: An efficient method for Bayesian inversion of multi-Gaussian log-hydraulic conductivity fields. *Water Resources Research*, *56*(8), e2020WR027110. https://doi.org/10.1029/2020WR027110

Yin, Y., Guen, V., Dona, J., Ayed, I., de Bézenac, E., Thome, N., & Gallinari, P. (2020). Augmenting physical models with deep networks for complex dynamics forecasting. *arXiv preprint arXiv:2010.04456*. https://doi.org/10.48550/arXiv.2010.04456

Zhuang, J., Kochkov, D., Bar-Sinai, Y., Brenner, M., & Hoyer, S. (2021). Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, *6*(6), 064605. https://doi.org/10.1103/physrevfluids.6.064605

Institut für Wasser- und Umweltsystemmodellierung
Universität Stuttgart

Pfaffenwaldring 61
70569 Stuttgart (Vaihingen)
Telefon (0711) 685 - 60156
Telefax (0711) 685 - 51073
E-Mail: iws@iws.uni-stuttgart.de
http://www.iws.uni-stuttgart.de

Direktoren
Prof. Dr.-Ing. Rainer Helmig
Prof. Dr.-Ing. Wolfgang Nowak
Prof. Dr.-Ing. Silke Wieprecht

Emeriti
Prof. Dr.-Ing. habil. Dr.-Ing. E.h. Jürgen Giesecke
Prof. Dr.h.c. Dr.-Ing. E.h. Helmut Kobus, PhD

Lehrstuhl für Wasserbau und Wassermengenwirtschaft
Leiterin: Prof. Dr.-Ing. Silke Wieprecht
Stellv.: Dr.-Ing. Kristina Terheiden
Versuchsanstalt für Wasserbau
Leiter: Stefan Haun, PhD

Lehrstuhl für Hydromechanik und Hydrosystemmodellierung
Leiter: Prof. Dr.-Ing. Rainer Helmig
Stellv.: apl. Prof. Dr.-Ing. Holger Class

Lehrstuhl für Stochastische Simulation und Sicherheitsforschung für Hydrosysteme
Leiter: Prof. Dr.-Ing. Wolfgang Nowak
Stellv.: apl. Prof. Dr.-Ing. Sergey Oladyshkin
Hydrogeophysik der Vadosen Zone
(mit Forschungszentrum Jülich)
Leiter: Prof. Dr. J.A. Sander Huisman

***VEGAS,* Versuchseinrichtung zur Grundwasser- und Altlastensanierung**
Leiter: Dr.-Ing. Simon Kleinknecht
PD Dr.-Ing. Claus Haslauer

Verzeichnis der Mitteilungshefte

7 *Sonderheft anläßlich des 65. Geburtstages von Prof. Arthur Röhnisch mit Beiträgen von* Benk, Dieter; Breitling, J.; Gurr, Siegfried; Haberhauer, Robert; Honekamp, Hermann; Kuz, Klaus Dieter; Marotz, Günter; Mayer-Vorfelder, Hans-Jörg; Miller, Rudolf; Plate, Erich J.; Radomski, Helge; Schwarz, Helmut; Vollmer, Ernst; Wildenhahn, Eberhard; 1967

8 Jumikis, Alfred: *Beitrag zur experimentellen Untersuchung des Wassernachschubs in einem gefrierenden Boden und die Beurteilung der Ergebnisse*, 1968

9 Marotz, Günter: *Technische Grundlagen einer Wasserspeicherung im natürlichen Untergrund*, 1968

10 Radomski, Helge: *Untersuchungen über den Einfluß der Querschnittsform wellenförmiger Spundwände auf die statischen und rammtechnischen Eigenschaften*, 1968

11 Schwarz, Helmut: *Die Grenztragfähigkeit des Baugrundes bei Einwirkung vertikal gezogener Ankerplatten als zweidimensionales Bruchproblem*, 1969

12 Erbel, Klaus: *Ein Beitrag zur Untersuchung der Metamorphose von Mittelgebirgsschneedecken unter besonderer Berücksichtigung eines Verfahrens zur Bestimmung der thermischen Schneequalität*, 1969

13 Westhaus, Karl-Heinz: *Der Strukturwandel in der Binnenschiffahrt und sein Einfluß auf den Ausbau der Binnenschiffskanäle*, 1969

14 Mayer-Vorfelder, Hans-Jörg: *Ein Beitrag zur Berechnung des Erdwiderstandes unter Ansatz der logarithmischen Spirale als Gleitflächenfunktion*, 1970

15 Schulz, Manfred: *Berechnung des räumlichen Erddruckes auf die Wandung kreiszylindrischer Körper*, 1970

16 Mobasseri, Manoutschehr: *Die Rippenstützmauer. Konstruktion und Grenzen ihrer Standsicherheit*, 1970

17 Benk, Dieter: *Ein Beitrag zum Betrieb und zur Bemessung von Hochwasserrückhaltebecken*, 1970

18 Gàl, Attila: *Bestimmung der mitschwingenden Wassermasse bei überströmten Fischbauchklappen mit kreiszylindrischem Staublech*, 1971, *vergriffen*

19 Kuz, Klaus Dieter: *Ein Beitrag zur Frage des Einsetzens von Kavitationserscheinungen in einer Düsenströmung bei Berücksichtigung der im Wasser gelösten Gase*, 1971, *vergriffen*

20 Schaak, Hartmut: *Verteilleitungen von Wasserkraftanlagen*, 1971

21 *Sonderheft zur Eröffnung der neuen Versuchsanstalt des Instituts für Wasserbau der Universität Stuttgart mit Beiträgen von* Brombach, Hansjörg; Dirksen, Wolfram; Gàl, Attila; Gerlach, Reinhard; Giesecke, Jürgen; Holthoff, Franz-Josef; Kuz, Klaus Dieter; Marotz, Günter; Minor, Hans-Erwin; Petrikat, Kurt; Röhnisch, Arthur; Rueff, Helge; Schwarz, Helmut; Vollmer, Ernst; Wildenhahn, Eberhard; 1972

22 Wang, Chung-su: *Ein Beitrag zur Berechnung der Schwingungen an Kegelstrahlschiebern*, 1972

23 Mayer-Vorfelder, Hans-Jörg: *Erdwiderstandsbeiwerte nach dem Ohde-Variationsverfahren*, 1972

24 Minor, Hans-Erwin: *Beitrag zur Bestimmung der Schwingungsanfachungsfunktionen überströmter Stauklappen*, 1972, *vergriffen*

25 Brombach, Hansjörg: *Untersuchung strömungsmechanischer Elemente (Fluidik) und die Möglichkeit der Anwendung von Wirbelkammerelementen im Wasserbau*, 1972, *vergriffen*

26 Wildenhahn, Eberhard: *Beitrag zur Berechnung von Horizontalfilterbrunnen*, 1972

27 Steinlein, Helmut: *Die Eliminierung der Schwebstoffe aus Flußwasser zum Zweck der unterirdischen Wasserspeicherung, gezeigt am Beispiel der Iller*, 1972

28 Holthoff, Franz Josef: *Die Überwindung großer Hubhöhen in der Binnenschiffahrt durch Schwimmerhebewerke*, 1973

29 Röder, Karl: *Einwirkungen aus Baugrundbewegungen auf trog- und kastenförmige Konstruktionen des Wasser- und Tunnelbaues*, 1973

30 Kretschmer, Heinz: *Die Bemessung von Bogenstaumauern in Abhängigkeit von der Talform*, 1973

31 Honekamp, Hermann: *Beitrag zur Berechnung der Montage von Unterwasserpipelines*, 1973

32 Giesecke, Jürgen: *Die Wirbelkammertriode als neuartiges Steuerorgan im Wasserbau*, und Brombach, Hansjörg: *Entwicklung, Bauformen, Wirkungsweise und Steuereigenschaften von Wirbelkammerverstärkern*, 1974

33 Rueff, Helge: *Untersuchung der schwingungserregenden Kräfte an zwei hintereinander angeordneten Tiefschützen unter besonderer Berücksichtigung von Kavitation*, 1974

34 Röhnisch, Arthur: *Einpreßversuche mit Zementmörtel für Spannbeton - Vergleich der Ergebnisse von Modellversuchen mit Ausführungen in Hüllwellrohren*, 1975

35 *Sonderheft anläßlich des 65. Geburtstages von Prof. Dr.-Ing. Kurt Petrikat mit Beiträgen von*: Brombach, Hansjörg; Erbel, Klaus; Flinspach, Dieter; Fischer jr., Richard; Gàl, Attila; Gerlach, Reinhard; Giesecke, Jürgen; Haberhauer, Robert; Hafner Edzard; Hausenblas, Bernhard; Horlacher, Hans-Burkhard; Hutarew, Andreas; Knoll, Manfred; Krummet, Ralph; Marotz, Günter; Merkle, Theodor; Miller, Christoph; Minor, Hans-Erwin; Neumayer, Hans; Rao, Syamala; Rath, Paul; Rueff, Helge; Ruppert, Jürgen; Schwarz, Wolfgang; Topal-Gökceli, Mehmet; Vollmer, Ernst; Wang, Chung-su; Weber, Hans-Georg; 1975

36 Berger, Jochum: *Beitrag zur Berechnung des Spannungszustandes in rotationssymmetrisch belasteten Kugelschalen veränderlicher Wandstärke unter Gas- und Flüssigkeitsdruck durch Integration schwach singulärer Differentialgleichungen*, 1975

37 Dirksen, Wolfram: *Berechnung instationärer Abflußvorgänge in gestauten Gerinnen mittels Differenzenverfahren und die Anwendung auf Hochwasserrückhaltebecken*, 1976

38 Horlacher, Hans-Burkhard: *Berechnung instationärer Temperatur- und Wärmespannungsfelder in langen mehrschichtigen Hohlzylindern*, 1976

39 Hafner, Edzard: *Untersuchung der hydrodynamischen Kräfte auf Baukörper im Tiefwasserbereich des Meeres*, 1977, ISBN 3-921694-39-6

40 Ruppert, Jürgen: *Über den Axialwirbelkammerverstärker für den Einsatz im Wasserbau*, 1977, ISBN 3-921694-40-X

41 Hutarew, Andreas: *Beitrag zur Beeinflußbarkeit des Sauerstoffgehalts in Fließgewässern an Abstürzen und Wehren*, 1977, ISBN 3-921694-41-8, *vergriffen*

42 Miller, Christoph: *Ein Beitrag zur Bestimmung der schwingungserregenden Kräfte an unterströmten Wehren*, 1977, ISBN 3-921694-42-6

43 Schwarz, Wolfgang: *Druckstoßberechnung unter Berücksichtigung der Radial- und Längsverschiebungen der Rohrwandung*, 1978, ISBN 3-921694-43-4

44 Kinzelbach, Wolfgang: *Numerische Untersuchungen über den optimalen Einsatz variabler Kühlsysteme einer Kraftwerkskette am Beispiel Oberrhein*, 1978, ISBN 3-921694-44-2

45 Barczewski, Baldur: *Neue Meßmethoden für Wasser-Luftgemische und deren Anwendung auf zweiphasige Auftriebsstrahlen*, 1979, ISBN 3-921694-45-0

46 Neumayer, Hans: *Untersuchung der Strömungsvorgänge in radialen Wirbelkammerverstärkern*, 1979, ISBN 3-921694-46-9

47 Elalfy, Youssef-Elhassan: *Untersuchung der Strömungsvorgänge in Wirbelkammerdioden und -drosseln*, 1979, ISBN 3-921694-47-7

48 Brombach, Hansjörg: *Automatisierung der Bewirtschaftung von Wasserspeichern*, 1981, ISBN 3-921694-48-5

49 Geldner, Peter: *Deterministische und stochastische Methoden zur Bestimmung der Selbstdichtung von Gewässern*, 1981, ISBN 3-921694-49-3, *vergriffen*

50 Mehlhorn, Hans: *Temperaturveränderungen im Grundwasser durch Brauchwassereinleitungen*, 1982, ISBN 3-921694-50-7, *vergriffen*

51 Hafner, Edzard: *Rohrleitungen und Behälter im Meer*, 1983, ISBN 3-921694-51-5

52 Rinnert, Bernd: *Hydrodynamische Dispersion in porösen Medien: Einfluß von Dichteunterschieden auf die Vertikalvermischung in horizontaler Strömung*, 1983, ISBN 3-921694-52-3, *vergriffen*

53 Lindner, Wulf: *Steuerung von Grundwasserentnahmen unter Einhaltung ökologischer Kriterien*, 1983, ISBN 3-921694-53-1, *vergriffen*

54 Herr, Michael; Herzer, Jörg; Kinzelbach, Wolfgang; Kobus, Helmut; Rinnert, Bernd: *Methoden zur rechnerischen Erfassung und hydraulischen Sanierung von Grundwasserkontaminationen*, 1983, ISBN 3-921694-54-X

55 Schmitt, Paul: *Wege zur Automatisierung der Niederschlagsermittlung*, 1984, ISBN 3-921694-55-8, *vergriffen*

56 Müller, Peter: *Transport und selektive Sedimentation von Schwebstoffen bei gestautem Abfluß*, 1985, ISBN 3-921694-56-6

57 El-Qawasmeh, Fuad: *Möglichkeiten und Grenzen der Tropfbewässerung unter besonderer Berücksichtigung der Verstopfungsanfälligkeit der Tropfelemente*, 1985, ISBN 3-921694-57-4, *vergriffen*

58 Kirchenbaur, Klaus: *Mikroprozessorgesteuerte Erfassung instationärer Druckfelder am Beispiel seegangsbelasteter Baukörper*, 1985, ISBN 3-921694-58-2

59 Kobus, Helmut (Hrsg.): *Modellierung des großräumigen Wärme- und Schadstofftransports im Grundwasser*, Tätigkeitsbericht 1984/85 (DFG-Forschergruppe an den Universitäten Hohenheim, Karlsruhe und Stuttgart), 1985, ISBN 3-921694-59-0, *vergriffen*

60 Spitz, Karlheinz: *Dispersion in porösen Medien: Einfluß von Inhomogenitäten und Dichteunterschieden*, 1985, ISBN 3-921694-60-4, *vergriffen*

61 Kobus, Helmut: *An Introduction to Air-Water Flows in Hydraulics*, 1985, ISBN 3-921694-61-2

62 Kaleris, Vassilios: *Erfassung des Austausches von Oberflächen- und Grundwasser in horizontalebenen Grundwassermodellen*, 1986, ISBN 3-921694-62-0

63 Herr, Michael: *Grundlagen der hydraulischen Sanierung verunreinigter Porengrundwasserleiter*, 1987, ISBN 3-921694-63-9

64 Marx, Walter: *Berechnung von Temperatur und Spannung in Massenbeton infolge Hydratation*, 1987, ISBN 3-921694-64-7

65 Koschitzky, Hans-Peter: *Dimensionierungskonzept für Sohlbelüfter in Schußrinnen zur Vermeidung von Kavitationsschäden*, 1987, ISBN 3-921694-65-5

66 Kobus, Helmut (Hrsg.): *Modellierung des großräumigen Wärme- und Schadstofftransports im Grundwasser*, Tätigkeitsbericht 1986/87 (DFG-Forschergruppe an den Universitäten Hohenheim, Karlsruhe und Stuttgart) 1987, ISBN 3-921694-66-3

67 Söll, Thomas: *Berechnungsverfahren zur Abschätzung anthropogener Temperaturanomalien im Grundwasser*, 1988, ISBN 3-921694-67-1

68 Dittrich, Andreas; Westrich, Bernd: *Bodenseeufererosion, Bestandsaufnahme und Bewertung*, 1988, ISBN 3-921694-68-X, *vergriffen*

69 Huwe, Bernd; van der Ploeg, Rienk R.: *Modelle zur Simulation des Stickstoffhaushaltes von Standorten mit unterschiedlicher landwirtschaftlicher Nutzung*, 1988, ISBN 3-921694-69-8, *vergriffen*

70 Stephan, Karl: *Integration elliptischer Funktionen*, 1988, ISBN 3-921694-70-1

71 Kobus, Helmut; Zilliox, Lothaire (Hrsg.): *Nitratbelastung des Grundwassers, Auswirkungen der Landwirtschaft auf die Grundwasser- und Rohwasserbeschaffenheit und Maßnahmen zum Schutz des Grundwassers*. Vorträge des deutsch-französischen Kolloquiums am 6. Oktober 1988, Universitäten Stuttgart und Louis Pasteur Strasbourg (Vorträge in deutsch oder französisch, Kurzfassungen zweisprachig), 1988, ISBN 3-921694-71-X

72 Soyeaux, Renald: *Unterströmung von Stauanlagen auf klüftigem Untergrund unter Berücksichtigung laminarer und turbulenter Fließzustände*,1991, ISBN 3-921694-72-8

73 Kohane, Roberto: *Berechnungsmethoden für Hochwasserabfluß in Fließgewässern mit überströmten Vorländern*, 1991, ISBN 3-921694-73-6

74 Hassinger, Reinhard: *Beitrag zur Hydraulik und Bemessung von Blocksteinrampen in flexibler Bauweise*, 1991, ISBN 3-921694-74-4, *vergriffen*

75 Schäfer, Gerhard: *Einfluß von Schichtenstrukturen und lokalen Einlagerungen auf die Längsdispersion in Porengrundwasserleitern*, 1991, ISBN 3-921694-75-2

76 Giesecke, Jürgen: *Vorträge, Wasserwirtschaft in stark besiedelten Regionen; Umweltforschung mit Schwerpunkt Wasserwirtschaft*, 1991, ISBN 3-921694-76-0

77 Huwe, Bernd: *Deterministische und stochastische Ansätze zur Modellierung des Stickstoffhaushalts landwirtschaftlich genutzter Flächen auf unterschiedlichem Skalenniveau*, 1992, ISBN 3-921694-77-9, *vergriffen*

78 Rommel, Michael: *Verwendung von Kluftdaten zur realitätsnahen Generierung von Kluftnetzen mit anschließender laminar-turbulenter Strömungsberechnung*, 1993, ISBN 3-92 1694-78-7

79 Marschall, Paul: *Die Ermittlung lokaler Stofffrachten im Grundwasser mit Hilfe von Einbohrloch-Meßverfahren*, 1993, ISBN 3-921694-79-5, *vergriffen*

80 Ptak, Thomas: *Stofftransport in heterogenen Porenaquiferen: Felduntersuchungen und stochastische Modellierung*, 1993, ISBN 3-921694-80-9, *vergriffen*

81 Haakh, Frieder: *Transientes Strömungsverhalten in Wirbelkammern,* 1993, ISBN 3-921694-81-7

82 Kobus, Helmut; Cirpka, Olaf; Barczewski, Baldur; Koschitzky, Hans-Peter: *Versuchseinrichtung zur Grundwasser- und Altlastensanierung VEGAS, Konzeption und Programmrahmen*, 1993, ISBN 3-921694-82-5

83 Zang, Weidong: *Optimaler Echtzeit-Betrieb eines Speichers mit aktueller Abflußregenerierung*, 1994, ISBN 3-921694-83-3, *vergriffen*

84 Franke, Hans-Jörg: *Stochastische Modellierung eines flächenhaften Stoffeintrages und Transports in Grundwasser am Beispiel der Pflanzenschutzmittelproblematik*, 1995, ISBN 3-921694-84-1

85 Lang, Ulrich: *Simulation regionaler Strömungs- und Transportvorgänge in Karstaquiferen mit Hilfe des Doppelkontinuum-Ansatzes: Methodenentwicklung und Parameteridenti-fikation*, 1995, ISBN 3-921694-85-X, *vergriffen*

86 Helmig, Rainer: *Einführung in die Numerischen Methoden der Hydromechanik*, 1996, ISBN 3-921694-86-8, *vergriffen*

87 Cirpka, Olaf: *CONTRACT: A Numerical Tool for Contaminant Transport and Chemical Transformations - Theory and Program Documentation -*, 1996, ISBN 3-921694-87-6

88 Haberlandt, Uwe: *Stochastische Synthese und Regionalisierung des Niederschlages für Schmutzfrachtberechnungen*, 1996, ISBN 3-921694-88-4

89 Croisé, Jean: *Extraktion von flüchtigen Chemikalien aus natürlichen Lockergesteinen mittels erzwungener Luftströmung*, 1996, ISBN 3-921694-89-2, *vergriffen*

90 Jorde, Klaus: *Ökologisch begründete, dynamische Mindestwasserregelungen bei Ausleitungskraftwerken*, 1997, ISBN 3-921694-90-6, *vergriffen*

91 Helmig, Rainer: *Gekoppelte Strömungs- und Transportprozesse im Untergrund - Ein Beitrag zur Hydrosystemmodellierung-*, 1998, ISBN 3-921694-91-4, *vergriffen*

92 Emmert, Martin: *Numerische Modellierung nichtisothermer Gas-Wasser Systeme in porösen Medien,* 1997, ISBN 3-921694-92-2

93 Kern, Ulrich: *Transport von Schweb- und Schadstoffen in staugeregelten Fließgewässern am Beispiel des Neckars*, 1997, ISBN 3-921694-93-0, *vergriffen*

94 Förster, Georg: *Druckstoßdämpfung durch große Luftblasen in Hochpunkten von Rohrleitungen 1997*, ISBN 3-921694-94-9

95 Cirpka, Olaf: *Numerische Methoden zur Simulation des reaktiven Mehrkomponententransports im Grundwasser*, 1997, ISBN 3-921694-95-7, *vergriffen*

96 Färber, Arne: *Wärmetransport in der ungesättigten Bodenzone: Entwicklung einer thermischen In-situ-Sanierungstechnologie*, 1997, ISBN 3-921694-96-5

97 Betz, Christoph: *Wasserdampfdestillation von Schadstoffen im porösen Medium: Entwicklung einer thermischen In-situ-Sanierungstechnologie*, 1998, SBN 3-921694-97-3

98 Xu, Yichun: *Numerical Modeling of Suspended Sediment Transport in Rivers*, 1998, ISBN 3-921694-98-1, *vergriffen*

99 Wüst, Wolfgang: *Geochemische Untersuchungen zur Sanierung CKW-kontaminierter Aquifere mit Fe(0)-Reaktionswänden,* 2000, ISBN 3-933761-02-2

100 Sheta, Hussam: *Simulation von Mehrphasenvorgängen in porösen Medien unter Einbeziehung von Hysterese-Effekten*, 2000, ISBN 3-933761-03-4

101 Ayros, Edwin: *Regionalisierung extremer Abflüsse auf der Grundlage statistischer Verfahren*, 2000, ISBN 3-933761-04-2, *vergriffen*

102 Huber, Ralf: *Compositional Multiphase Flow and Transport in Heterogeneous Porous Media*, 2000, ISBN 3-933761-05-0

103 Braun, Christopherus: *Ein Upscaling-Verfahren für Mehrphasenströmungen in porösen Medien*, 2000, ISBN 3-933761-06-9

104 Hofmann, Bernd: *Entwicklung eines rechnergestützten Managementsystems zur Beurteilung von Grundwasserschadensfällen*, 2000, ISBN 3-933761-07-7

105 Class, Holger: *Theorie und numerische Modellierung nichtisothermer Mehrphasenprozesse in NAPL-kontaminierten porösen Medien,* 2001, ISBN 3-933761-08-5

106 Schmidt, Reinhard: *Wasserdampf- und Heißluftinjektion zur thermischen Sanierung kontaminierter Standorte*, 2001, ISBN 3-933761-09-3

107 Josef, Reinhold: *Schadstoffextraktion mit hydraulischen Sanierungsverfahren unter Anwendung von grenzflächenaktiven Stoffen*, 2001, ISBN 3-933761-10-7

108 Schneider, Matthias: *Habitat- und Abflussmodellierung für Fließgewässer mit unscharfen Berechnungsansätzen*, 2001, ISBN 3-933761-11-5

109 Rathgeb, Andreas: *Hydrodynamische Bemessungsgrundlagen für Lockerdeckwerke an überströmbaren Erddämmen*, 2001, ISBN 3-933761-12-3

110 Lang, Stefan: *Parallele numerische Simulation instätionärer Probleme mit adaptiven Methoden auf unstrukturierten Gittern*, 2001, ISBN 3-933761-13-1

111 Appt, Jochen; Stumpp Simone: *Die Bodensee-Messkampagne 2001, IWS/CWR Lake Constance Measurement Program 2001*, 2002, ISBN 3-933761-14-X

112 Heimerl, Stephan: *Systematische Beurteilung von Wasserkraftprojekten,* 2002, ISBN 3-933761-15-8, *vergriffen*

113 Iqbal, Amin: *On the Management and Salinity Control of Drip Irrigation,* 2002, ISBN 3-933761-16-6

114 Silberhorn-Hemminger, Annette: *Modellierung von Kluftaquifersystemen: Geostatistische Analyse und deterministisch-stochastische Kluftgenerierung,* 2002, ISBN 3-933761-17-4

115 Winkler, Angela: *Prozesse des Wärme- und Stofftransports bei der In-situ-Sanierung mit festen Wärmequellen,* 2003, ISBN 3-933761-18-2

116 Marx, Walter: *Wasserkraft, Bewässerung, Umwelt - Planungs- und Bewertungsschwerpunkte der Wasserbewirtschaftung,* 2003, ISBN 3-933761-19-0

117 Hinkelmann, Reinhard: *Efficient Numerical Methods and Information-Processing Techniques in Environment Water,* 2003, ISBN 3-933761-20-4

118 Samaniego-Eguiguren, Luis Eduardo: *Hydrological Consequences of Land Use / Land Cover and Climatic Changes in Mesoscale Catchments*, 2003, ISBN 3-933761-21-2

119 Neunhäuserer, Lina: *Diskretisierungsansätze zur Modellierung von Strömungs- und Transportprozessen in geklüftet-porösen Medien*, 2003, ISBN 3-933761-22-0

120 Paul, Maren: *Simulation of Two-Phase Flow in Heterogeneous Poros Media with Adaptive Methods*, 2003, ISBN 3-933761-23-9

121 Ehret, Uwe: *Rainfall and Flood Nowcasting in Small Catchments using Weather Radar*, 2003, ISBN 3-933761-24-7

122 Haag, Ingo: *Der Sauerstoffhaushalt staugeregelter Flüsse am Beispiel des Neckars - Analysen, Experimente, Simulationen -,* 2003, ISBN 3-933761-25-5

123 Appt, Jochen: *Analysis of Basin-Scale Internal Waves in Upper Lake Constance,* 2003, ISBN 3-933761-26-3

124 Hrsg.: Schrenk, Volker; Batereau, Katrin; Barczewski, Baldur; Weber, Karolin und Koschitzky, Hans-Peter: *Symposium Ressource Fläche und VEGAS - Statuskolloquium 2003, 30. September und 1. Oktober 2003,* 2003, ISBN 3-933761-27-1

125 Omar Khalil Ouda: *Optimisation of Agricultural Water Use: A Decision Support System for the Gaza Strip,* 2003, ISBN 3-933761-28-0

126 Batereau, Katrin: *Sensorbasierte Bodenluftmessung zur Vor-Ort-Erkundung von Schadensherden im Untergrund*, 2004, ISBN 3-933761-29-8

127 Witt, Oliver: *Erosionsstabilität von Gewässersedimenten mit Auswirkung auf den Stofftransport bei Hochwasser am Beispiel ausgewählter Stauhaltungen des Oberrheins*, 2004, ISBN 3-933761-30-1

128 Jakobs, Hartmut: *Simulation nicht-isothermer Gas-Wasser-Prozesse in komplexen Kluft-Matrix-Systemen*, 2004, ISBN 3-933761-31-X

129 Li, Chen-Chien: *Deterministisch-stochastisches Berechnungskonzept zur Beurteilung der Auswirkungen erosiver Hochwasserereignisse in Flussstauhaltungen,* 2004, ISBN 3-933761-32-8

130 Reichenberger, Volker; Helmig, Rainer; Jakobs, Hartmut; Bastian, Peter; Niessner, Jennifer: *Complex Gas-Water Processes in Discrete Fracture-Matrix Systems: Up-scaling, Mass-Conservative Discretization and Efficient Multilevel Solution,* 2004, ISBN 3-933761-33-6

131 Hrsg.: Barczewski, Baldur; Koschitzky, Hans-Peter; Weber, Karolin; Wege, Ralf: *VEGAS - Statuskolloquium 2004,* Tagungsband zur Veranstaltung am 05. Oktober 2004 an der Universität Stuttgart, Campus Stuttgart-Vaihingen, 2004, ISBN 3-933761-34-4

132 Asie, Kemal Jabir: *Finite Volume Models for Multiphase Multicomponent Flow through Porous Media*. 2005, ISBN 3-933761-35-2

133 Jacoub, George: *Development of a 2-D Numerical Module for Particulate Contaminant Transport in Flood Retention Reservoirs and Impounded Rivers,* 2004, ISBN 3-933761-36-0

134 Nowak, Wolfgang: *Geostatistical Methods for the Identification of Flow and Transport Parameters in the Subsurface,* 2005, ISBN 3-933761-37-9

135 Süß, Mia: *Analysis of the influence of structures and boundaries on flow and transport processes in fractured porous media*, 2005, ISBN 3-933761-38-7

136 Jose, Surabhin Chackiath: *Experimental Investigations on Longitudinal Dispersive Mixing in Heterogeneous Aquifers,* 2005, ISBN: 3-933761-39-5

137 Filiz, Fulya: *Linking Large-Scale Meteorological Conditions to Floods in Mesoscale Catchments,* 2005, ISBN 3-933761-40-9

138 Qin, Minghao: *Wirklichkeitsnahe und recheneffiziente Ermittlung von Temperatur und Spannungen bei großen RCC-Staumauern*, 2005, ISBN 3-933761-41-7

139 Kobayashi, Kenichiro: *Optimization Methods for Multiphase Systems in the Subsurface - Application to Methane Migration in Coal Mining Areas*, 2005, ISBN 3-933761-42-5

140 Rahman, Md. Arifur: *Experimental Investigations on Transverse Dispersive Mixing in Heterogeneous Porous Media,* 2005, ISBN 3-933761-43-3

141 Schrenk, Volker: *Ökobilanzen zur Bewertung von Altlastensanierungsmaßnahmen,* 2005, ISBN 3-933761-44-1

142 Hundecha, Hirpa Yeshewatesfa: *Regionalization of Parameters of a Conceptual Rainfall-Runoff Model*, 2005, ISBN: 3-933761-45-X

143 Wege, Ralf: *Untersuchungs- und Überwachungsmethoden für die Beurteilung natürlicher Selbstreinigungsprozesse im Grundwasser,* 2005, ISBN 3-933761-46-8

144 Breiting, Thomas: *Techniken und Methoden der Hydroinformatik - Modellierung von komplexen Hydrosystemen im Untergrund*, 2006, ISBN 3-933761-47-6

145 Hrsg.: Braun, Jürgen; Koschitzky, Hans-Peter; Müller, Martin: *Ressource Untergrund: 10 Jahre VEGAS: Forschung und Technologieentwicklung zum Schutz von Grundwasser und Boden,* Tagungsband zur Veranstaltung am 28. und 29. September 2005 an der Universität Stuttgart, Campus Stuttgart-Vaihingen, 2005, ISBN 3-933761-48-4

146 Rojanschi, Vlad: *Abflusskonzentration in mesoskaligen Einzugsgebieten unter Berücksichtigung des Sickerraumes,* 2006, ISBN 3-933761-49-2

147 Winkler, Nina Simone: *Optimierung der Steuerung von Hochwasserrückhaltebeckensystemen,* 2006, ISBN 3-933761-50-6

148 Wolf, Jens: *Räumlich differenzierte Modellierung der Grundwasserströmung alluvialer Aquifere für mesoskalige Einzugsgebiete*, 2006, ISBN: 3-933761-51-4

149 Kohler, Beate: *Externe Effekte der Laufwasserkraftnutzung*, 2006, ISBN 3-933761-52-2

150 Hrsg.: Braun, Jürgen; Koschitzky, Hans-Peter; Stuhrmann, Matthias: *VEGAS-Statuskolloquium 2006,* Tagungsband zur Veranstaltung am 28. September 2006 an der Universität Stuttgart, Campus Stuttgart-Vaihingen, 2006, ISBN 3-933761-53-0

151 Niessner, Jennifer: *Multi-Scale Modeling of Multi-Phase - Multi-Component Processes in Heterogeneous Porous Media*, 2006, ISBN 3-933761-54-9

152 Fischer, Markus: *Beanspruchung eingeerdeter Rohrleitungen infolge Austrocknung bindiger Böden,* 2006, ISBN 3-933761-55-7

153 Schneck, Alexander: *Optimierung der Grundwasserbewirtschaftung unter Berücksichtigung der Belange der Wasserversorgung, der Landwirtschaft und des Naturschutzes*, 2006, ISBN 3-933761-56-5

154 Das, Tapash: *The Impact of Spatial Variability of Precipitation on the Predictive Uncertainty of Hydrological Models,* 2006, ISBN 3-33761-57-3

155 Bielinski, Andreas: *Numerical Simulation of CO_2 sequestration in geological formations*, 2007, ISBN 3-933761-58-1

156 Mödinger, Jens: *Entwicklung eines Bewertungs- und Entscheidungsunterstützungssystems für eine nachhaltige regionale Grundwasserbewirtschaftung*, 2006, ISBN 3-933761-60-3

157 Manthey, Sabine: *Two-phase flow processes with dynamic effects in porous media - parameter estimation and simulation,* 2007, ISBN 3-933761-61-1

158 Pozos Estrada, Oscar: *Investigation on the Effects of Entrained Air in Pipelines,* 2007, ISBN 3-933761-62-X

159 Ochs, Steffen Oliver: *Steam injection into saturated porous media – process analysis including experimental and numerical investigations*, 2007, ISBN 3-933761-63-8

160 Marx, Andreas: *Einsatz gekoppelter Modelle und Wetterradar zur Abschätzung von Niederschlagsintensitäten und zur Abflussvorhersage, 2007,* ISBN 3-933761-64-6

161 Hartmann, Gabriele Maria: *Investigation of Evapotranspiration Concepts in Hydrological Modelling for Climate Change Impact Assessment*, 2007, ISBN 3-933761-65-4

162 Kebede Gurmessa, Tesfaye: *Numerical Investigation on Flow and Transport Characteristics to Improve Long-Term Simulation of Reservoir Sedimentation*, 2007, ISBN 3-933761-66-2

163 Trifković, Aleksandar: *Multi-objective and Risk-based Modelling Methodology for Planning, Design and Operation of Water Supply Systems*, 2007, ISBN 3-933761-67-0

164 Götzinger, Jens: *Distributed Conceptual Hydrological Modelling - Simulation of Climate, Land Use Change Impact and Uncertainty Analysis*, 2007, ISBN 3-933761-68-9

165 Hrsg.: Braun, Jürgen; Koschitzky, Hans-Peter; Stuhrmann, Matthias: *VEGAS – Kolloquium 2007,* Tagungsband zur Veranstaltung am 26. September 2007 an der Universität Stuttgart, Campus Stuttgart-Vaihingen, 2007, ISBN 3-933761-69-7

166 Freeman, Beau: *Modernization Criteria Assessment for Water Resources Planning; Klamath Irrigation Project, U.S.*, 2008, ISBN 3-933761-70-0

167 Dreher, Thomas: *Selektive Sedimentation von Feinstschwebstoffen in Wechselwirkung mit wandnahen turbulenten Strömungsbedingungen*, 2008, ISBN 3-933761-71-9

168 Yang, Wei: *Discrete-Continuous Downscaling Model for Generating Daily Precipitation Time Series*, 2008, ISBN 3-933761-72-7

169 Kopecki, Ianina: *Calculational Approach to FST-Hemispheres for Multiparametrical Benthos Habitat Modelling,* 2008*,* ISBN 3-933761-73-5

170 Brommundt, Jürgen: *Stochastische Generierung räumlich zusammenhängender Niederschlagszeitreihen*, 2008, ISBN 3-933761-74-3

171 Papafotiou, *Alexandros: Numerical Investigations of the Role of Hysteresis in Heterogeneous Two-Phase Flow Systems*, 2008, ISBN 3-933761-75-1

172 He, Yi: *Application of a Non-Parametric Classification Scheme to Catchment Hydrology*, 2008, ISBN 978-3-933761-76-7

173 Wagner, Sven: *Water Balance in a Poorly Gauged Basin in West Africa Using Atmospheric Modelling and Remote Sensing Information*, 2008, ISBN 978-3-933761-77-4

174 Hrsg.: Braun, Jürgen; Koschitzky, Hans-Peter; Stuhrmann, Matthias; Schrenk, Volker: *VEGAS-Kolloquium 2008 Ressource Fläche III*, Tagungsband zur Veranstaltung am 01. Oktober 2008 an der Universität Stuttgart, Campus Stuttgart-Vaihingen, 2008, ISBN 978-3-933761-78-1

175 Patil, Sachin: *Regionalization of an Event Based Nash Cascade Model for Flood Predictions in Ungauged Basins*, 2008, ISBN 978-3-933761-79-8

176 Assteerawatt, Anongnart: *Flow and Transport Modelling of Fractured Aquifers based on a Geostatistical Approach,* 2008, ISBN 978-3-933761-80-4

177 Karnahl, Joachim Alexander: *2D numerische Modellierung von multifraktionalem Schwebstoff- und Schadstofftransport in Flüssen*, 2008, ISBN 978-3-933761-81-1

178 Hiester, Uwe: *Technologieentwicklung zur In-situ-Sanierung der ungesättigten Bodenzone mit festen Wärmequellen,* 2009, ISBN 978-3-933761-82-8

179 Laux, Patrick: *Statistical Modeling of Precipitation for Agricultural Planning in the Volta Basin of West Africa,* 2009, ISBN 978-3-933761-83-5

180 Ehsan, Saqib: *Evaluation of Life Safety Risks Related to Severe Flooding,* 2009, ISBN 978-3-933761-84-2

181 Prohaska, Sandra: *Development and Application of a 1D Multi-Strip Fine Sediment Transport Model for Regulated Rivers,* 2009, ISBN 978-3-933761-85-9

182 Kopp, Andreas: *Evaluation of CO_2 Injection Processes in Geological Formations for Site Screening,* 2009, ISBN 978-3-933761-86-6

183 Ebigbo, Anozie: *Modelling of biofilm growth and its influence on CO_2 and water (two-phase) flow in porous media,* 2009, ISBN 978-3-933761-87-3

184 Freiboth, Sandra: *A phenomenological model for the numerical simulation of multiphase multicomponent processes considering structural alterations of porous media*, 2009, ISBN 978-3-933761-88-0

185 Zöllner, Frank: *Implementierung und Anwendung netzfreier Methoden im Konstruktiven Wasserbau und in der Hydromechanik*, 2009, ISBN 978-3-933761-89-7

186 Vasin, Milos: *Influence of the soil structure and property contrast on flow and transport in the unsaturated zone*, 2010, ISBN 978-3-933761-90-3

187 Li, Jing: *Application of Copulas as a New Geostatistical Tool*, 2010, ISBN 978-3-933761-91-0

188 AghaKouchak, Amir: *Simulation of Remotely Sensed Rainfall Fields Using Copulas*, 2010, ISBN 978-3-933761-92-7

189 Thapa, Pawan Kumar: *Physically-based spatially distributed rainfall runoff modelling for soil erosion estimation*, 2010, ISBN 978-3-933761-93-4

190 Wurms, Sven: *Numerische Modellierung der Sedimentationsprozesse in Retentionsanlagen zur Steuerung von Stoffströmen bei extremen Hochwasserabflussereignissen*, 2011, ISBN 978-3-933761-94-1

191 Merkel, Uwe: *Unsicherheitsanalyse hydraulischer Einwirkungen auf Hochwasserschutzdeiche und Steigerung der Leistungsfähigkeit durch adaptive Strömungsmodellierung*, 2011, ISBN 978-3-933761-95-8

192 Fritz, Jochen: *A Decoupled Model for Compositional Non-Isothermal Multiphase Flow in Porous Media and Multiphysics Approaches for Two-Phase Flow*, 2010, ISBN 978-3-933761-96-5

193 Weber, Karolin (Hrsg.): *12. Treffen junger WissenschaftlerInnen an Wasserbauinstituten*, 2010, ISBN 978-3-933761-97-2

194 Bliefernicht, Jan-Geert: *Probability Forecasts of Daily Areal Precipitation for Small River Basins,* 2011, ISBN 978-3-933761-98-9

195 Hrsg.: Koschitzky, Hans-Peter; Braun, Jürgen: *VEGAS-Kolloquium 2010 In-situ-Sanierung - Stand und Entwicklung Nano und ISCO -,* Tagungsband zur Veranstaltung am 07. Oktober 2010 an der Universität Stuttgart, Campus Stuttgart-Vaihingen*,* 2010*,* ISBN 978-3-933761-99-6

196 Gafurov, Abror: *Water Balance Modeling Using Remote Sensing Information - Focus on Central Asia,* 2010, ISBN 978-3-942036-00-9

197 Mackenberg, Sylvia: *Die Quellstärke in der Sickerwasserprognose: Möglichkeiten und Grenzen von Labor- und Freilanduntersuchungen,* 2010, ISBN 978-3-942036-01-6

198 Singh, Shailesh Kumar: *Robust Parameter Estimation in Gauged and Ungauged Basins,* 2010, ISBN 978-3-942036-02-3

199 Doğan, Mehmet Onur: *Coupling of porous media flow with pipe flow,* 2011, ISBN 978-3-942036-03-0

200 Liu, Min: *Study of Topographic Effects on Hydrological Patterns and the Implication on Hydrological Modeling and Data Interpolation,* 2011, ISBN 978-3-942036-04-7

201 Geleta, Habtamu Itefa: *Watershed Sediment Yield Modeling for Data Scarce Areas,* 2011, ISBN 978-3-942036-05-4

202 Franke, Jörg: *Einfluss der Überwachung auf die Versagenswahrscheinlichkeit von Staustufen,* 2011, ISBN 978-3-942036-06-1

203 Bakimchandra, Oinam: *Integrated Fuzzy-GIS approach for assessing regional soil erosion risks,* 2011, ISBN 978-3-942036-07-8

204 Alam, Muhammad Mahboob: *Statistical Downscaling of Extremes of Precipitation in Mesoscale Catchments from Different RCMs and Their Effects on Local Hydrology,* 2011, ISBN 978-3-942036-08-5

205 Hrsg.: Koschitzky, Hans-Peter; Braun, Jürgen: *VEGAS-Kolloquium 2011 Flache Geothermie - Perspektiven und Risiken,* Tagungsband zur Veranstaltung am 06. Oktober 2011 an der Universität Stuttgart, Campus Stuttgart-Vaihingen*,* 2011*,* ISBN 978-3-933761-09-2

206 Haslauer, Claus: *Analysis of Real-World Spatial Dependence of Subsurface Hydraulic Properties Using Copulas with a Focus on Solute Transport Behaviour,* 2011, ISBN 978-3-942036-10-8

207 Dung, Nguyen Viet: *Multi-objective automatic calibration of hydrodynamic models – development of the concept and an application in the Mekong Delta,* 2011, ISBN 978-3-942036-11-5

208 Hung, Nguyen Nghia: *Sediment dynamics in the floodplain of the Mekong Delta, Vietnam,* 2011, ISBN 978-3-942036-12-2

209 Kuhlmann, Anna: *Influence of soil structure and root water uptake on flow in the unsaturated zone,* 2012, ISBN 978-3-942036-13-9

210 Tuhtan, Jeffrey Andrew: *Including the Second Law Inequality in Aquatic Ecodynamics: A Modeling Approach for Alpine Rivers Impacted by Hydropeaking,* 2012, ISBN 978-3-942036-14-6

211 Tolossa, Habtamu: *Sediment Transport Computation Using a Data-Driven Adaptive Neuro-Fuzzy Modelling Approach,* 2012, ISBN 978-3-942036-15-3

212 Tatomir, Alexandru-Bodgan: *From Discrete to Continuum Concepts of Flow in Fractured Porous Media,* 2012, ISBN 978-3-942036-16-0

213 Erbertseder, Karin: *A Multi-Scale Model for Describing Cancer-Therapeutic Transport in the Human Lung,* 2012, ISBN 978-3-942036-17-7

214 Noack, Markus: *Modelling Approach for Interstitial Sediment Dynamics and Reproduction of Gravel Spawning Fish,* 2012, ISBN 978-3-942036-18-4

215 De Boer, Cjestmir Volkert: *Transport of Nano Sized Zero Valent Iron Colloids during Injection into the Subsurface*, 2012, ISBN 978-3-942036-19-1

216 Pfaff, Thomas: *Processing and Analysis of Weather Radar Data for Use in Hydrology*, 2013, ISBN 978-3-942036-20-7

217 Lebrenz, Hans-Henning: *Addressing the Input Uncertainty for Hydrological Modeling by a New Geostatistical Method*, 2013, ISBN 978-3-942036-21-4

218 Darcis, Melanie Yvonne: *Coupling Models of Different Complexity for the Simulation of CO_2 Storage in Deep Saline Aquifers*, 2013, ISBN 978-3-942036-22-1

219 Beck, Ferdinand: *Generation of Spatially Correlated Synthetic Rainfall Time Series in High Temporal Resolution - A Data Driven Approach*, 2013, ISBN 978-3-942036-23-8

220 Guthke, Philipp: *Non-multi-Gaussian spatial structures: Process-driven natural genesis, manifestation, modeling approaches, and influences on dependent processes*, 2013, ISBN 978-3-942036-24-5

221 Walter, Lena: *Uncertainty studies and risk assessment for CO_2 storage in geological formations*, 2013, ISBN 978-3-942036-25-2

222 Wolff, Markus: *Multi-scale modeling of two-phase flow in porous media including capillary pressure effects,* 2013, ISBN 978-3-942036-26-9

223 Mosthaf, Klaus Roland: *Modeling and analysis of coupled porous-medium and free flow with application to evaporation processes,* 2014, ISBN 978-3-942036-27-6

224 Leube, Philipp Christoph: *Methods for Physically-Based Model Reduction in Time: Analysis, Comparison of Methods and Application*, 2013, ISBN 978-3-942036-28-3

225 Rodríguez Fernández, Jhan Ignacio: *High Order Interactions among environmental variables: Diagnostics and initial steps towards modeling*, 2013, ISBN 978-3-942036-29-0

226 Eder, Maria Magdalena: *Climate Sensitivity of a Large Lake*, 2013, ISBN 978-3-942036-30-6

227 Greiner, Philipp: *Alkoholinjektion zur In-situ-Sanierung von CKW Schadensherden in Grundwasserleitern: Charakterisierung der relevanten Prozesse auf unterschiedlichen Skalen*, 2014, ISBN 978-3-942036-31-3

228 Lauser, Andreas: *Theory and Numerical Applications of Compositional Multi-Phase Flow in Porous Media*, 2014, ISBN 978-3-942036-32-0

229 Enzenhöfer, Rainer: *Risk Quantification and Management in Water Production and Supply Systems*, 2014, ISBN 978-3-942036-33-7

230 Faigle, Benjamin: *Adaptive modelling of compositional multi-phase flow with capillary pressure*, 2014, ISBN 978-3-942036-34-4

231 Oladyshkin, Sergey: *Efficient modeling of environmental systems in the face of complexity and uncertainty*, 2014, ISBN 978-3-942036-35-1

232 Sugimoto, Takayuki: *Copula based Stochastic Analysis of Discharge Time Series*, 2014, ISBN 978-3-942036-36-8

233 Koch, Jonas: *Simulation, Identification and Characterization of Contaminant Source Architectures in the Subsurface*, 2014, ISBN 978-3-942036-37-5

234 Zhang, Jin: *Investigations on Urban River Regulation and Ecological Rehabilitation Measures, Case of Shenzhen in China*, 2014, ISBN 978-3-942036-38-2

235 Siebel, Rüdiger: *Experimentelle Untersuchungen zur hydrodynamischen Belastung und Standsicherheit von Deckwerken an überströmbaren Erddämmen,* 2014, ISBN 978-3-942036-39-9

236 Baber, Katherina: *Coupling free flow and flow in porous media in biological and technical applications: From a simple to a complex interface description,* 2014, ISBN 978-3-942036-40-5

237 Nuske, Klaus Philipp: *Beyond Local Equilibrium — Relaxing local equilibrium assumptions in multiphase flow in porous media,* 2014, ISBN 978-3-942036-41-2

238 Geiges, Andreas: *Efficient concepts for optimal experimental design in nonlinear environmental systems*, 2014, ISBN 978-3-942036-42-9

239 Schwenck, Nicolas: *An XFEM-Based Model for Fluid Flow in Fractured Porous Media*, 2014, ISBN 978-3-942036-43-6

240 Chamorro Chávez, Alejandro: *Stochastic and hydrological modelling for climate change prediction in the Lima region, Peru*, 2015, ISBN 978-3-942036-44-3

241 Yulizar: *Investigation of Changes in Hydro-Meteorological Time Series Using a Depth-Based Approach*, 2015, ISBN 978-3-942036-45-0

242 Kretschmer, Nicole: *Impacts of the existing water allocation scheme on the Limarí watershed – Chile, an integrative approach*, 2015, ISBN 978-3-942036-46-7

243 Kramer, Matthias: *Luftbedarf von Freistrahlturbinen im Gegendruckbetrieb*, 2015, ISBN 978-3-942036-47-4

244 Hommel, Johannes: *Modeling biogeochemical and mass transport processes in the subsurface: Investigation of microbially induced calcite precipitation*, 2016, ISBN 978-3-942036-48-1

245 Germer, Kai: *Wasserinfiltration in die ungesättigte Zone eines makroporösen Hanges und deren Einfluss auf die Hangstabilität*, 2016, ISBN 978-3-942036-49-8

246 Hörning, Sebastian: *Process-oriented modeling of spatial random fields using copulas*, 2016, ISBN 978-3-942036-50-4

247 Jambhekar, Vishal: *Numerical modeling and analysis of evaporative salinization in a coupled free-flow porous-media system,* 2016, ISBN 978-3-942036-51-1

248 Huang, Yingchun: *Study on the spatial and temporal transferability of conceptual hydrological models*, 2016, ISBN 978-3-942036-52-8

249 Kleinknecht, Simon Matthias: *Migration and retention of a heavy NAPL vapor and remediation of the unsaturated zone*, 2016, ISBN 978-3-942036-53-5

250 Kwakye, Stephen Oppong: *Study on the effects of climate change on the hydrology of the West African sub-region*, 2016, ISBN 978-3-942036-54-2

251 Kissinger, Alexander: *Basin-Scale Site Screening and Investigation of Possible Impacts of CO_2 Storage on Subsurface Hydrosystems*, 2016, ISBN 978-3-942036-55-9

252 Müller, Thomas: *Generation of a Realistic Temporal Structure of Synthetic Precipitation Time Series for Sewer Applications*, 2017, ISBN 978-3-942036-56-6

253 Grüninger, Christoph: *Numerical Coupling of Navier-Stokes and Darcy Flow for Soil-Water Evaporation*, 2017, ISBN 978-3-942036-57-3

254 Suroso: *Asymmetric Dependence Based Spatial Copula Models: Empirical Investigations and Consequences on Precipitation Fields*, 2017, ISBN 978-3-942036-58-0

255 Müller, Thomas; Mosthaf, Tobias; Gunzenhauser, Sarah; Seidel, Jochen; Bárdossy, András: *Grundlagenbericht Niederschlags-Simulator (NiedSim3)*, 2017, ISBN 978-3-942036-59-7

256 Mosthaf, Tobias: *New Concepts for Regionalizing Temporal Distributions of Precipitation and for its Application in Spatial Rainfall Simulation,* 2017, ISBN 978-3-942036-60-3

257 Fenrich, Eva Katrin: *Entwicklung eines ökologisch-ökonomischen Vernetzungsmodells für Wasserkraftanlagen und Mehrzweckspeicher,* 2018, ISBN 978-3-942036-61-0

258 Schmidt, Holger: *Microbial stabilization of lotic fine sediments*, 2018, ISBN 978-3-942036-62-7

259 Fetzer, Thomas: *Coupled Free and Porous-Medium Flow Processes Affected by Turbulence and Roughness – Models, Concepts and Analysis*, 2018, ISBN 978-3-942036-63-4

260 Schröder, Hans Christoph: *Large-scale High Head Pico Hydropower Potential Assessment*, 2018, ISBN 978-3-942036-64-1

261 Bode, Felix: *Early-Warning Monitoring Systems for Improved Drinking Water Resource Protection*, 2018, ISBN 978-3-942036-65-8

262 Gebler, Tobias: *Statistische Auswertung von simulierten Talsperrenüberwachungsdaten zur Identifikation von Schadensprozessen an Gewichtsstaumauern*, 2018, ISBN 978-3-942036-66-5

263 Harten, Matthias von: *Analyse des Zuppinger-Wasserrades – Hydraulische Optimierungen unter Berücksichtigung ökologischer Aspekte*, 2018, ISBN 978-3-942036-67-2

264 Yan, Jieru: *Nonlinear estimation of short time precipitation using weather radar and surface observations*, 2018, ISBN 978-3-942036-68-9

265 Beck, Martin: *Conceptual approaches for the analysis of coupled hydraulic and geomechanical processes*, 2019, ISBN 978-3-942036-69-6

266 Haas, Jannik: *Optimal planning of hydropower and energy storage technologies for fully renewable power systems*, 2019, ISBN 978-3-942036-70-2

267 Schneider, Martin: *Nonlinear Finite Volume Schemes for Complex Flow Processes and Challenging Grids*, 2019, ISBN 978-3-942036-71-9

268 Most, Sebastian Christopher: *Analysis and Simulation of Anomalous Transport in Porous Media*, 2019, ISBN 978-3-942036-72-6

269 Buchta, Rocco: *Entwicklung eines Ziel- und Bewertungssystems zur Schaffung nachhaltiger naturnaher Strukturen in großen sandgeprägten Flüssen des norddeutschen Tieflandes*, 2019, ISBN 978-3-942036-73-3

270 Thom, Moritz: *Towards a Better Understanding of the Biostabilization Mechanisms of Sediment Beds*, 2019, ISBN 978-3-942036-74-0

271 Stolz, Daniel: *Die Nullspannungstemperatur in Gewichtsstaumauern unter Berücksichtigung der Festigkeitsentwicklung des Betons*, 2019, ISBN 978-3-942036-75-7

272 Rodriguez Pretelin, Abelardo: *Integrating transient flow conditions into groundwater well protection*, 2020, ISBN: 978-3-942036-76-4

273 Weishaupt, Kilian: *Model Concepts for Coupling Free Flow with Porous Medium Flow at the Pore-Network Scale: From Single-Phase Flow to Compositional Non-Isothermal Two-Phase Flow*, 2020, ISBN: 978-3-942036-77-1

274 Koch, Timo: *Mixed-dimension models for flow and transport processes in porous media with embedded tubular network systems*, 2020, ISBN: 978-3-942036-78-8

275 Gläser, Dennis: *Discrete fracture modeling of multi-phase flow and deformation in fractured poroelastic media*, 2020, ISBN: 978-3-942036-79-5

276 Seitz, Lydia: *Development of new methods to apply a multi-parameter approach – A first step towards the determination of colmation*, 2020, ISBN: 978-3-942036-80-1

277 Ebrahim Bakhshipour, Amin: *Optimizing hybrid decentralized systems for sustainable ur-ban drainage infrastructures planning*, 2021, ISBN: 978-3-942036-81-8

278 Seitz, Gabriele: *Modeling Fixed-Bed Reactors for Thermochemical Heat Storage with the Reaction System $CaO/Ca(OH)_2$*, 2021, ISBN: 978-3-942036-82-5

279 Emmert, Simon: *Developing and Calibrating a Numerical Model for Microbially Enhanced Coal-Bed Methane Production*, 2021, ISBN: 978-3-942036-83-2

280 Heck, Katharina Klara: *Modelling and analysis of multicomponent transport at the interface between free- and porous-medium flow - influenced by radiation and roughness*, 2021, ISBN: 978-3-942036-84-9

281 Ackermann, Sina: *A multi-scale approach for drop/porous-medium interaction*, 2021, ISBN: 978-3-942036-85-6

282 Beckers, Felix: *Investigations on Functional Relationships between Cohesive Sediment Erosion and Sediment Characteristics*, 2021, ISBN: 978-3-942036-86-3

283 Schlabing, Dirk: *Generating Weather for Climate Impact Assessment on Lakes*, 2021, ISBN: 978-3-942036-87-0

284 Becker, Beatrix: *Efficient multiscale multiphysics models accounting for reversible flow at various subsurface energy storage sites*, 2021, ISBN: 978-3-942036-88-7

285 Reuschen, Sebastian: *Bayesian Inversion and Model Selection of Heterogeneities in Geo-statistical Subsurface Modeling*, 2021, ISBN: 978-3-942036-89-4

286 Michalkowski, Cynthia: *Modeling water transport at the interface between porous GDL and gas distributor of a PEM fuel cell cathode*, 2022, ISBN: 978-3-942036-90-0

287 Koca, Kaan: *Advanced experimental methods for investigating flow-biofilm-sediment interactions*, 2022, ISBN: 978-3-942036-91-7

288 Modiri, Ehsan: *Clustering simultaneous occurrences of extreme floods in the Neckar catchment*, 2022, ISBN: 978-3-942036-92-4

289 Mayar, Mohammad Assem: *High-resolution spatio-temporal measurements of the colmation phenomenon under laboratory conditions*, 2022, ISBN: 978-3-942036-93-1

290 Schäfer Rodrigues Silva, Aline: *Quantifying and Visualizing Model Similarities for Multi-Model Methods*, 2022, ISBN: 978-3-942036-94-8

291 Moreno Leiva, Simón: *Optimal planning of water and renewable energy systems for copper production processes with sector coupling and demand flexibility*, 2022, ISBN 978-3-942036-95-5

292 Schönau, Steffen: *Modellierung von Bodenerosion und Sedimentaustrag bei Hochwasserereignissen am Beispiel des Einzugsgebiets der Rems*, 2022, ISBN 978-3-942036-96-2

293 Glatz, Kumiko: *Upscaling of Nanoparticle Transport in Porous Media,* 2022, ISBN 978-3-942036-97-9

294 Pavía Santolamazza, Daniela: *Event-based flood estimation using a random forest algorithm for the regionalization in small catchments*, 2022, ISBN 978-3-942036-98-6

295 Haun, Stefan: *Advanced Methods for a Sustainable Sediment Management of Reservoirs*, 2022, ISBN 978-3-942036-99-3

296 Herma, Felix: *Data Processing and Model Choice for Flood Prediction*, 2022, ISBN 978-3-910293-00-7

297 Weinhardt, Felix: *Porosity and permeability alterations in processes of biomineralization in porous media - microfluidic investigations and their interpretation*, 2022, ISBN 978-3-910293-01-4

298 Sadid, Najibullah: *Bedload Transport Estimation in Mountainous Intermittent Rivers and Streams*, 2023, ISBN 978-3-910293-02-1

299 Mohammadi, Farid: *A Surrogate-Assisted Bayesian Framework for Uncertainty-Aware Validation Benchmarks*, 2023, ISBN 978-3-910293-03-8

300 Praditia, Timothy: *Physics-informed Neural Networks for Learning Dynamic, Distributed and Uncertain Systems*, 2023, ISBN 978-3-910293-04-5

Die Mitteilungshefte ab der Nr. 134 (Jg. 2005) stehen als pdf-Datei über die Homepage des Instituts: www.iws.uni-stuttgart.de zur Verfügung.