MODELLING PHYSICS-BASED DYNAMIC SYSTEM USING MACHINE LEARNING

By

Tianshu Bao

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

August 11, 2023

Nashville, Tennessee

Approved:

Taylor Thomas Johnson, Ph.D.

Janos Sztipanovits, Ph.D.

Tyler Derr, Ph.D.

Xiaowei Jia, Ph.D.

Yuankai Huo, Ph.D.

ACKNOWLEDGMENTS

Time flies. I can't believe I have already spent 6 years in Nashville from my late 20s to early 30s. There are many people I would like to thank. Firstly, I would like to express my sincere gratitude to my adviser Prof. Taylor Johnson for the continuous support of my PhD study and related research, and for his patience and immense knowledge. Besides my advisor, I would like to thank Prof. Xiaowei Jia. His guidance helped me in all the time of research and writing of this dissertation. Because of his assistance and mentoring, I have had a chance to publish my work at top conferences and step into the physics-guided machine learning research field. I also thank my thesis committee: Prof. Janos Sztipanovits, Prof. Tyler Derr and Prof. Yuankai Huo, for taking the time to serve on my dissertation committee and providing insightful comments and suggestions, which incented me to widen my research from various perspectives. In particular, I am thankful to Prof. Weiming Xiang, and Prof. Hoang Dung Tran, who gave me suggestions for verification research work. I also would like to thank my colleague, Shengyu Chen, who is a graduate student from the University of Pittsburgh and worked closely with me on the projects, and my lab mates, for the cherished time spent together over the last six years. I would also thank Dr Louise Hanson from Student Health Center, for her kindness and help during my PhD study. Finally, I want to take this opportunity to express my profound gratitude to my parents, Weimin Chen and Lei Bao, my cousin, Yingying Cao, and her husband for their encouragement and endless love.

Acknowledgement of Support

The material presented is based upon work supported by the National Science Foundation (NSF) through Grant 2028001, Grant OAC-2203581, awards 1910017, 1918450, and 2028001, the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-18-C-0089, the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019, the USGS Award G21AC10207, and Pitt Momentum Award. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF.

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 Motivation

Cyber-Physical Systems (CPS) have the potential to revolutionize the development of technology worldwide, and in recent years, numerous researchers and companies have invested significant time and money into ensuring that these systems will realize their economic and societal potential [114]. However since CPS fundamentally integrate computational and physical processes, there are unique challenges in designing and analyzing these systems. In fact, the functional correctness of CPS relies deeply on the dynamics of their physical environment and the discrete control decisions of their computational units [104]. Within hybrid systems, the framework of Hybrid Automata (HA) has demonstrated considerable utility in capturing the complex interaction between the discrete and continuous parts of a CPS. Additionally, it allows for a formal analysis about the safety and reliability of CPS [104] where safety and reliability are two key properties of CPS. Safety is aimed at protecting the systems from accidental failures in order to avoid hazards, while reliability is focused on performing consistently well. Safety verification of CPS is the act of proving or disproving the correctness of safety properties and reachability analysis is one of the fundamental problems of it. Reachability analysis computes the set of possible solutions of dynamical systems subject to uncertain initial states and inputs. Classically, this modeling framework has catered to systems with dynamics that can be described by ordinary differential equations (ODEs), and hardly any attention has been paid to systems with dynamics described by partial differential equations (PDEs) [189].

PDEs have been widely used to describe a wide range of phenomena such as fluid dynamics and quantum mechanics that cannot be adequately described by ODEs [58]. As an example, a typical application for PDEs is the modeling of congestion in highway networks [19], and one popular model for highway control is the Lighthill-Whitham-Richards (LWR) model [118, 132]. Finite different methods (FDMs) have been widely used to solve PDEs [175]. They convert PDE into a system of linear equations that can be solved by matrix algebra techniques and are one of the most common approaches to the numerical solution of PDE. The truncation error of FDMs is defined as the difference between the approximation and the exact analytical solution and is represented in the form of Big-O notation. However, these constants before the notation are always unobtainable which forces us to seek a better way to solve PDEs more accurately.

Since classical numerical methods express their limitation dealing with PDEs, we could utilize the promising machine learning (ML) methods as a powerful tool towards our problem. There is no surprise that ma-

chine learning methods are increasingly used in CPSs to perform different difficult tasks. Machine learning models, which have found immense success in commercial applications, are beginning to play an important role in advancing scientific discovery [76, 165, 167]. Given their power in automatically learning from observation data, ML models are particularly promising in scientific problems involving complex processes that are not completely understood by our current body of knowledge. However, scientific problems often involve non-stationary relationships among physical variables which can change over space and time. In the absence of adequate information about the physical mechanisms of real-world processes, traditional ML approaches are prone to false discoveries because it is difficult to capture these complex relationships solely from data. Therefore, it is necessary to combine physical based approaches with ML that can be used to simulate and verify PDE driven systems.

1.2 Research Challenges

Since many CPSs involve sensing and controlling PDE-modeled physical phenomena, reachability analysis of PDEs should provide novel methods for safety verification and falsification. However, due to the fact that the solutions of most PDEs are not obtainable, it forces us to seek an estimated reachability analysis for the original problem. In our work, reachability analysis for PDEs are done through FDM, interpolation, optimization and parallelization. And the construction of reachable sets are based on the properties of PDEs and these approaches. However, since FDM can only provide an estimated solution, it would be better to look at other more accurate methods. For the optimization step appearing at nonlinear problem, the accuracy of the output relies heavily on the choice of the optimizer. Thus this remains to be a topic needing further discussion.

Instead of using classical methods, a promising way is to use ML approach. In scientific domains, physics-based models are often used to study engineering and environmental systems. Even though these models are based on known physical laws that govern relationships between input and output variables, most physics based models are necessarily approximations of reality due to incomplete knowledge of certain processes or omission of processes to maintain computational efficiency. In fact, there are numerous sophisticated PDEs developed to describe nature phenomena. They are foundational in the modern scientific understanding of sound, heat, diffusion, electrodynamics, fluid dynamics and quantum mechanics. In particular, existing physics-based approaches for predicting river networks simulate the internal distribution of target variables (e.g. temperature) based on general physical relationships such as energy and mass conservation. These conservation laws are critical parts constructing the fluid dynamic and are always expressed as PDEs since PDEs give the spatial and temporal relations between different physical variables such as density, velocity, mass and etc. However, the model predictions still rely on qualitative parameterization (approximations)

based on soil and surficial geologic classification along with topography, land cover and climate input. Hence, such models can only provide sub-optimal prediction performance.

A classical application of PDE is fluid dynamics. Traditional fluid models have used the Navier–Stokes equation [18] for simulating fluid dynamics in many applications including aquatic science, hydraulic modeling, weather and climate modeling, ocean currents, and aerodynamics. When modeling temperature dynamics in river networks, these PDEs capture not only the temporal thermodynamics but also the spatial heat diffusion and convection from connected river segments [54]. Furthermore, these PDEs, along with other known physical relationships, have been used to build more complex physics-based models [124] to simulate multiple interacting processes on different variables in a system. However, these equations and physics-based models have limits in their predictions due to approximations and parameterization used to represent underlying processes.

Recent advances in ML, given their great success in commercial applications, have provided unrealized potential for modeling complex data patterns in scientific problems. The power of these models come from their capacity to extract complex nonlinear patterns from observation data and naturally incorporate spatial and temporal data dependencies. For example, recurrent neural networks (RNN)-based models, which take account of temporal dependencies, have shown extensive applicability in speech recognition and machine translation. Convolutional neural network (CNN)-based approaches have shown tremendous success in learning spatial patterns in many computer vision applications. Recently, graph neural network models, e.g., graph convolutional networks (GCN), have shown a great promise for modeling interactions and similarities amongst multiple objects and also have shown encouraging results for studying river networks.

However, there still exist several challenges that can not be handled properly by traditional ML model. Accuracy of neural network methods can be a problem when simulating physical process. Physical behavior may change suddenly under certain conditions, and traditional neural network may not be able to capture the subtle change due to the large time step. We need a physics-based ML model can simulate the real physical relation according to its underline physical laws. Meanwhile, we show that our model performs much better than existing models, but it remains limited in precisely predicting special segments due to unobserved areas. Models are always specific and restricted to particular systems. Extending the current model to a general framework could also be an interesting topic. The general framework can serve as pre-trained model to any specific river network.

1.3 Contributions

In this section, we present the contributions towards addressing the outlined challenges of modelling real-world scientific problems. The main contributions are listed below.

1.3.1 Chapter 3

This chapter presents the syntax, semantics, and decidability results of a new type of HA with partial differential equation dynamic, partial differential hybrid automata (PDHA), whose continuous dynamics are described by partial differential equations. While classically the dynamics of HA are described by ODEs and differential inclusions, PDHA are capable of describing the behavior of CPS with continuous dynamics that cannot be modelled using the canonical hybrid system framework. For the purposes of analyzing PDHA, we propose another model called the discrete space partial differential hybrid automata (DSPDHA) which handles discrete spatial domains. Additionally, we formally outline concepts such as time trajectories, PDHA and DSPDHA execution, Zeno behavior, among others to make thorough analysis convenient. We conclude with two illustrative examples in order to exhibit the nature of PDHA and DSPDHA.

1.3.2 Chapter 4

In this chapter, we consider the numerical reachability analysis for the linear/nonlinear hyperbolic equations with initial sets. The approaches proposed combine the flux splitting method, the Lax-Friedrich method, and the Lax-Wendroff method with optimization techniques for estimating reachable sets. Linear and nonlinear systems are analyzed based on our proposed methods. Interpolations are used to construct and bloat the reachable sets for the safety verification purpose. We show that our approaches indeed provide an asymptotic estimation of the real reachable sets with dense discretization.

Meanwhile, instead of hyperbolic equations, the numerical reachability analysis for one/two-dimension parabolic equations under dense spatial discretization is also analyzed. We propose new approaches that combine Krylov subspace method with numerical PDEs methods including Alternative direction implicit (ADI) method, Crank-Nicolson (C-N) method, and Alternative Segment Crank-Nicolson (ASC-N) method. We also harness the interpolations to construct and bloat the numerical reachable sets for the safety verification purpose. Several examples show that our approaches indeed provide an asymptotic estimation of the real reachable sets.

1.3.3 Chapter 5

In this chapter, we present a physics-guided machine learning approach that incorporates PDEs in a graph neural network model to improve the prediction of water temperature in river networks. The standard graph neural network model often uses pre-defined edge weights based on distance or similarity measures. Such static graph structure can be limited in capturing multiple processes in a physical system that interact and evolve over time. The limitation to represent underlying physical processes can severely impact the performance of the predictive model especially when we have access to limited training data. To better capture the

dynamic interactions among multiple segments in a river network, we built a dynamic graph model, where the graph structure is driven by the PDE that describes underlying physical processes. We further combine the dynamic graph structure and the recurrent layers to model temporal dependencies and improve the prediction. We demonstrate the effectiveness of the proposed method in a sub-network of the Delaware River Basin. In particular, we show that the proposed method outperforms existing physics-based and machine learning models in temperature prediction using sparse observation data for training. The proposed method has also been shown to produce better performance when generalized to different seasons.

1.3.4 Chapter 6

In this chapter, we propose a physics-guided neural network for reconstructing frequent DNS from sparse LES data by enhancing its spatial resolution and temporal frequency. Our proposed method consists of a PDE-based recurrent unit for capturing underlying temporal processes and a physics-guided super-resolution model that incorporates additional physical constraints. We demonstrate the effectiveness of both components in reconstructing the data generated by simulating the Taylor-Green Vortex sparse LES data. Moreover, we show that the proposed recurrent unit can preserve the physical characteristics of turbulent flows by leveraging the physical relationships in the Navier-Stokes equation.

1.3.5 Chapter 7

In this chapter, we propose a transfer learning approach under inaccurate physical rules, which can achieve robust regionalization performance under a gauged prediction scenario. We corrected the uncertain physical descriptors obtained through the physics rule by residual approximation and let these corrected descriptors rejoin the model training process. Our results show that the transfer learning model using our proposed residual approximation achieves a predictive performance comparable to that of the model using actual physical descriptors.

1.4 Copyright Acknowledgements

The required copyright statements for permission to reprint portions of [15, 17] are included in the following.

- For portions of [17] reproduced in this dissertation, we acknowledge the IEEE copyright: © 2021 IEEE. Reprinted, with permission, from Tianshu Bao, Xiaowei Jia, Jacob Zwart, Jeffrey Sadler, Alison Appling, Samantha Oliver and Taylor T. Johnson, “Partial Differential Equation Driven Dynamic Graph Networks for Predicting Stream Water Temperature,” IEEE International Conference on Data Mining (ICDM), Dec 2021

- For portions of [15] reproduced in this dissertation, we acknowledge the AUAI Press copyright: © 2022 the AUAI Press. Reprinted, with permission, from Tianshu Bao, Shengyu Chen, Taylor T Johnson, Peyman Givi, Shervin Sammak and Xiaowei Jia, "Physics Guided Neural Networks for Spatio-temporal Super-resolution of Turbulent Flows," The Conference on Uncertainty in Artificial Intelligence (UAI), Jun 2022.

CHAPTER 2

Related Work

2.1 Safety Assurance in Cyber-physical Systems

Safety of certain CPS relies on reachable set/state estimation results that are based on Lyapunov functions analogous to stability [202, 203, 204, 205, 208, 211, 218] and reachability analysis of dynamical systems [206, 207], certainly have potentials to be further extended to safety verification. One can study the stability of switched systems, a switched system is composed of a family of continuous or discrete-time subsystems along with a switching rule governing the switching between the subsystems, with the help of the given the switching rule described by a prescribed state space partitioning [93, 123, 143] or some known constraints on switching sequence such as dwell time [6, 130] or average dwell time [86, 217] restrictions.

2.1.1 Hybrid Automaton

A hybrid system is a dynamical system with both discrete and continuous components. For example, an automobile engine whose fuel injection (continuous) is regulated by a microprocessor (discrete) is a hybrid system. As embedded computing becomes ubiquitous, hybrid systems are increasingly employed in safety-critical applications, thus making reliability a prime concern. Rigorous reliability analysis requires formal modeling. For this purpose, the hybrid automaton has been proposed as a formal model for hybrid systems.

In automata theory, a hybrid automaton is a mathematical model for precisely describing systems in which digital computational processes interact with analog physical processes. A hybrid automaton is a finite state machine with a finite set of continuous variables whose values are described by a set of ordinary differential equations. This combined specification of discrete and continuous behaviors enables dynamic systems that comprise both digital and analog components to be modeled and analyzed.

2.1.2 Safety Verification of Cyber-Physical Systems with Reinforcement Learning Control

Safety verification of neural network control systems (NNCS) is a challenging problem because the behaviors of the systems are difficult to estimate or characterize. To explicitly analyze the safety of NNCS, the authors calculate the exact and overapproximate reachable set containing all possible trajectories of the plant that takes the control set from the neural network controller as inputs. The output set of the plant is feedback to the controller to compute the control set for the next control step. Therefore, if the error in the reachable set computation is large, it quickly becomes larger and larger over time which results in too conservative reachable sets that cannot be used for safety verification. In addition, the scalability and efficiency of the

reachable set computation are crucial for safety verification of control systems with DNN controllers. It is required methods that can compute the reachable set of NNCS with large neural network controllers with a reasonable computation time and a small over-approximation error. However, calculating an exact or tight, overapproximate reachable set of a neural network quickly is fundamentally difficult due to the non-linearity of the network. This challenging problem has not addressed well in the existing literature.

In this paper [184], the authors propose a new reachability analysis approach for safety verification of CPS with neural network controllers using the concept of star set. The authors particularly focus on the safety verification of the Advanced Emergency Braking System (AEBS) in an autonomous car to illustrate and evaluate the approach. The AEBS is controlled by a neural network controller which is trained to stop the vehicle appropriately if it discovers an obstacle on the road. To guarantee safety, it is required that the time-to-collision (TTC) of the car, which is a nonlinear function of the car's velocity, acceleration and the distance between the vehicle to the obstacle, is always larger than a safe threshold defined by the physical characteristics of the vehicle. The safety verification approach for AEBS works as follows. First, using CARLA, the authors perform system identification to obtain a discrete, linear state-space model of the car. The car model is then validated via systematic testing. Second, the authors train a deep neural network controller to perform the emergency braking action using reinforcement learning. Third, the authors compose the neural network controller with the state-space model to construct a closed-loop Simulink model of the AEBS which is then validated with CARLA results. Fourth, the authors perform the reachability analysis of the closed-loop model to obtain the reachable set of the AEBS. Finally, the authors compute the reachable set of the TTC and use it for safety verification.

The proposed reachability analysis approach is to feed forward neural network controllers with ReLU/Saturation activation functions. The reachability algorithms can compute both exact and over-approximate reachable sets of the AEBS. Exact reachable set computation is expensive since the number of the reachable sets increases over time steps. In contrast, the over-approximate reachability scheme is much cheaper, as it produces a single reachable set at each time step. Importantly, by using star sets, the reachability analysis approach can eliminate or reduce significantly the overapproximation errors which is the main reason that makes the obtained reachable sets more and more conservative over time as shown in the polyhedron approach [186, 206] (and maybe in some existing methods). The approach successfully verifies the safety of the AEBS and, notably, determines the entire region of the initial conditions of the AEBS where safety is guaranteed. This demonstrates the promising applicability of the approach in verifying the safety properties of neural network-based autonomous systems at design time. The polyhedron approach fails to prove the safety property of the system due to its over-approximation errors exploding quickly over time. In summary, the main contributions of this study are as follows. (1) the provision of star-based reachability schemes de-

signed to efficiently compute the reachable set of a discrete, linear neural network control systems with ReLU activation function, (2) an end-to-end design and implementation of these schemes in a MATLAB®toolbox called NNV [186] which is publicly available for evaluation and comparison, (3) and a thorough evaluation on the safety verification of the practical automatic emergency braking system.

In conclusion, the authors have proposed two efficient, exact and over-approximate reachability schemes and an optimization-based approach for safety verification of CPS with RL controller where the safety specification is defined based on a nonlinear transformation of the system states. From thorough experiments on the practical AEBS, the proposed method is computationally cheaper and less conservative than the existing polyhedron approach. More important, it is applicable to real-world applications.

2.2 Partial Differential Equations

PDEs have been widely used to describe a wide range of phenomena such as fluid dynamics and quantum mechanics that cannot be adequately described by ODEs [58]. As an example, a typical application for PDEs is the modeling of congestion in highway networks [19], and one popular model for highway control is the LWR model [118, 132]. Finite different methods have been widely used to solve PDEs [175].

2.3 Reachability Analysis

Reachability analysis is a solution to the reachability problem in the context of dynamical systems and is the fundamental problem in safety verification of CPS. It computes the set of possible solutions of dynamical systems subject to uncertain initial states and inputs. The problem of systems of linear ODEs has been studied extensively. Set representations include but are not limited to: zonotopes [8], support functions [73], and level sets [129]. Reachability analysis tools like SpaceEx [64] and Flow* [40], which utilize Taylor models, have proven to be powerful. Order reduction abstraction also plays a significant role in evaluating models. The first work combining reachability analysis with order reduction techniques is [81], and a more recent approach is discussed in [188].

Although there is significant work developing reachability analysis and verification methods for dynamical systems of ODEs and switched/hybrid extensions thereof, dynamical systems that utilize PDEs have not been widely investigated. PDEs are capable of describing phenomena such as fluid dynamics, quantum mechanics and digital signal processing [62] that ODEs do not adequately model. Reachability analysis for one-dimensional heat equations with ranged input and initial condition, known as linear parabolic equations, has previously been studied in [189]. In this context, the PDEs evolve in an infinite dimensional space and must be discretized for computational purposes [183].

2.3.1 Reachability Analysis for One Dimensional Linear Parabolic Equations

As many CPS involve sensing and control of physical phenomena modeled as PDEs, reachability analysis of PDEs provides novel methods for safety verification and falsification. As a first step to address this challenging problem, this paper proposes a reachability analysis approach leveraging the well-known Galerkin Finite Element Method (FEM) for a class of one-dimensional linear parabolic PDEs with fixed but uncertain inputs and initial conditions, which is a subclass of PDEs that is useful for modeling, for instance, heat flows. In particular, a continuous approximate reachable set of the parabolic PDE is computed using linear interpolation. Since a complete conservativeness is hardly achieved by using the approximate reachable set, to enhance the conservativeness, the authors investigate the error bound between the numerical solution and the exact analytically unsolvable solution to bloat the continuous approximate reachable set. This bloated reachable set is then used for safety verification and falsification. In the case that the safety specification is violated, the approach produces a numerical trace to prove that there exists an initial condition and input that lead the system to an unsafe state.

Although the heat equation has been demonstrated to be a good benchmark for accessing the scalability of verification techniques, a deeper study should be done for two reasons. Firstly, it is reasonable to have a safety specification concerned with a region in space and not only concentrated at specific mesh points. In other words, the authors are interested in continuous-space and not discrete-space reachability analysis. Second, it is crucial to have an approach that works for more general types of inputs and initial conditions, i.e., an input described by a nonlinear function in both time and space, and an initial condition defined by a nonlinear spatial function.

This work [189] proposes a continuous reachability analysis approach for linear parabolic equations with time invariant uncertain nonlinear inputs and initial conditions. By "continuous", it means that the continuity in both space and time are investigated. The main contributions are : 1) an extension of the well-known space-time Galerkin method and linear interpolation into a continuous reachability analysis approach for one dimensional parabolic equation; 2) enhancing the conservativeness of the proposed method by investigating and utilizing the error between the numerical solution and the exact analytically unsolvable solution; 3) providing an implementation the proposed method in a prototype called pdev, which is available online for further experimentation and evaluation.

In conclusion, a reachability analysis approach for linear parabolic equation is proposed based on the well-known Galerkin FEM. The conservativeness of the method is enhanced by utilizing the error caused by the Galerkin FEM to obtain a bloated continuous reachable set before using it to check the safety of a system. The evaluation section has shown that the method is practically applicable where the safety verifica-

tion/falsification problem can be solved efficiently with an appropriate computation cost. Moreover, the time complexity of the method is smaller than the traditional reachability analysis methods because the approach is simulation-equivalent.

2.3.2 Reachability Analysis for High-Index Linear Differential Algebraic Equations

Although many methods have been developed for reachability analysis of CPS, most of them mentioned above focus on CPS with ODE dynamics. There is a lack of methodology in analyzing systems with high-index DAE dynamics. It is because the reachability analysis for DAE systems is more complex than ODE systems, especially for high-index DAEs because they contain both a differential part (i.e., ODE) and algebraic constraints (AC). It should be emphasized that there are efficient reachability analysis approaches for DAE systems with index1 [9, 45, 48]. Dealing with index-1 DAE is slightly different from coping with pure ODE because, with a consistent initial condition, a semi-explicit index-1 DAE can be converted to an ODE. As CPS involving high-index DAE dynamics appear extensively in engineering and science such as multi-body mechanics, electrical circuit design, heat and gas transfer, chemical process, atmospheric physics, thermodynamic systems, and water distribution network [34, 57], there is an urgent need for novel reachability analysis methods and tools that can either verify or falsify the safety properties of such CPS. Solving this challenging problem is the main contribution of this research.

The novelty of the approach [187] comes from its objective in dealing with high index DAE which is a popular class of dynamics that has not been addressed in the existing literature. In this paper, the authors investigate the reachability analysis for large linear DAE systems with the index up to 3, which appear widely in practice. There are a variety of definitions for the index of a linear DAE. However, throughout the paper, the authors use the concept of tractability index proposed in [125] to determine the index of a linear DAE system. The approach consists of three main steps (a) decoupling and consistency checking, (b) reachable set computation, and (c) safety verification or falsification; that can be summarized as follows.

The first step is to use the Marz decoupling method [14, 125] to decouple a high index DAE into one ODE subsystem and one or several algebraic constraint (AC) subsystems. The core step in decoupling is constructing a set of admissible projectors which has not previously been discussed deeply in the existing literature. In this paper, the authors propose a novel algorithm that can construct such admissible projectors for a linear DAE system with the index up to 3 (most of DAE systems in practice have index from 1 to 3). Additionally, the authors define a consistent space for the DAE because, unlike ODE reachability analysis where the initial set of states can be freely defined by a user, to guarantee a numerical solution for the DAE system, the initial state and inputs of such DAE system must be consistent and satisfy certain constraints. It is important to emphasize that the decoupling and consistency checking methods used in the approach can

be combined with existing over-approximation reachability analysis methods [7, 64][2,19] to compute the over-approximated reachable sets for high-index, linear DAE systems with small to medium dimensions.

The second step in the approach is reachable set computation. Since the main objective is to verify or falsify large linear DAEs, the authors extend ODE simulation based reachability analysis to DAEs. In particular, the authors modify the generalized star-set proposed in [13] to enhance the efficiency in checking the initial condition consistency and safety for DAEs. From a consistent initial set of states and inputs, the reachable set of a DAE system can be constructed by combining the reachable sets of its subsystems. It is also worth pointing out that the piecewise constant inputs assumption for ODE with inputs used in [13] may lead a DAE system to impulsive behavior. Therefore, in this paper, the authors assume the inputs applied to the system are smooth functions. Such the inputs can be obtained by smoothing piecewise constant inputs with filters.

The last step in the approach is to verify or falsify the safety properties of the DAE system using the constructed reachable set computed in the second step. In this paper, the authors consider linear safety specifications. the authors are interested in checking the safety of the system in a specific direction defined using a directional matrix. Using the modified star-set and the directional matrix, checking the safety property can be solved efficiently as a low-dimensional feasibility linear programming problem. In the case of violation, the approach generates a counterexample trace that falsifies the system safety. The main contributions of the paper are as follows. 1. A novel reachability analysis approach for high-index linear DAE systems developed based on the effective combination of a decoupling method and a reachable set computation using star-set. To the best of the knowledge, this problem has not been addressed in the existing literature. 2. An end-to-end design and implementation of the approach in a Python toolbox, called Daev, which is publicly available for verifying high-index linear DAE systems. 3. An extensive evaluation that demonstrates the capability of the approach in verifying/falsifying a wide range of practical, high-index linear DAE systems where the number of state variables varies from several to thousands

2.3.3 Star-Based Reachability Analysis for Deep Neural Networks

In this paper [185], the authors propose a fast and scalable approach for the exact and over-approximate reachability analysis of DNN with ReLU activation functions using the concept of star sets [13], or shortly "star". Star fits perfectly for the reachability analysis of DNNs due to its following essential characteristics: 1) an efficient (exact) representation of large input sets; 2) fast and cheap affine mapping operations; 3) inexpensive intersections with half-spaces and checking empty. By utilizing star, the authors avoid the expensive affine mapping operation in polyhedron-based approach [23] and thus, reduce the verification time significantly. The approach performs reachability analysis for feedforward DNNs layer-by-layer. In the case

of exact analysis, the output reachable set of each layer is a union of a set of stars. Based on this observation, the star-based exact reachability algorithm naturally can be designed for efficient execution on multicore platforms where each layer can handle multiple input sets at the same time. In the case of over-approximate analysis, the output reachable set of each layer is a single star which can be constructed by doing point-wise over-approximation of the reachable set at all neurons of the layer.

The authors evaluate the proposed algorithms in comparison with the polyhedron approach [186], Reluplex [98], zonotope [171] and abstract domain [172] approaches on safety verification of the ACAS Xu neural networks [94] and robust certification of image classification DNN. The experimental results show that the exact reachability algorithm can achieve 19 times faster than Reluplex when running on multi-core platform and > 70 times faster than the polyhedron approach. Notably, the exact algorithm can visualize the precise behavior of the ACAS Xu networks and can construct the complete set of counter example inputs in the case that a safety property is violated. The over-approximate reachability algorithm is averagely 118 times faster than Reluplex. It successfully verifies many safety properties of ACAS Xu networks while the zonotope and abstract domain approaches fail due to their large over-approximation errors. the overapproximate reachability algorithm also provides a better robustness certification for image classification DNN in comparison with the zonotope and abstract domain approaches. In summary, the main contributions of this paper are: 1)propose novel, fast and scalable methods for the exact and over-approximate reachability analysis of DNNs; 2) implement the proposed methods in NNV toolbox that is available online for evaluation and comparison; 3) provide a thorough evaluation of the new methods via real-world case studies.

In conclusion, the authors proposed two reachability analysis algorithms for DNNs using star sets, one that is exact (sound and complete) but has scalability challenges and one that over-approximates (sound) with better scalability. The exact algorithm can compute and visualize the exact behaviors of DNNs. The exact method is more efficient than standard polyhedra approaches, and faster than SMT-based approaches when running on multi-core platforms. The over-approximate algorithm is much faster than the exact one, and notably, it is much less conservative than recent zonotope and abstract-domain based approaches. The algorithms are applicable for real world applications as shown in the safety verification of ACAS Xu DNNs and robustness certification of image classification DNNs.

2.4 Physics-Based Machine Learning

Physics-based models of dynamical systems are often used to study engineering and environmental systems. Despite their extensive use, these models have several well-known limitations due to incomplete or inaccurate representations of the physical processes being modeled. Given rapid data growth due to advances in sensor technologies, there is a tremendous opportunity to systematically advance modeling in these domains

by using machine learning (ML) methods. However, direct application of black-box ML models to a scientific problem encounters several major challenges. First, in the absence of adequate information about the physical mechanisms of real-world processes, ML approaches are prone to false discoveries and can also exhibit serious inconsistencies with known physics. This is because scientific problems often involve complex spaces of hypotheses with non-stationary relationships among the variables that are difficult to capture solely from the data. Second, black-box ML models suffer from poor interpretability since they are not explicitly designed for representing physical relationships and providing mechanistic insights. Third, the data available for several scientific problems are far smaller than what is needed to effectively train advanced ML models. Leveraging physics will be key to constrain hypothesis spaces to do ML in such small sample regimes. Hence, neither an ML-only nor a physics-only approach can be considered sufficient for knowledge discovery in complex scientific and engineering applications. Instead, there is a need to explore the continuum between physics-based and ML models, where both physics and data are integrated in a synergistic manner. Next the authorsoutline issues involved in building such a hybrid model that is already beginning to show great promise [50].

In science and engineering applications, a physical model often predicts values of many variables. Machine learning models can also generate predictions for many variables (e.g., by having multiple nodes in the output layer of a neural network). Most ML algorithms make use of a loss function that captures the difference between predicted and actual (i.e., observed values) to guide the search for parameter values that attempts to minimize this loss function. Although, such empirical models are often used in many scientific communities as alternatives to physical models, they fail to take into account many physical aspects of modeling. In the following, the authorslist some of these. In science and engineering applications, all errors (i.e., difference between predicted and observed values) may not be equally important. For example, for the lake temperature monitoring application, accuracy at surface and at high depth can be more important than error at the middle levels of the lake.

Instead of minimizing the difference between predicted and observed values, it may be more important to optimize the prediction of a different physical quantity, which can be computed from the observed or predicted values. For example, for certain lake temperature monitoring applications, the ability to correctly predict the depth of thermocline (i.e., the depth at which temperature gradient is maximum) can be more important than correctly predicting the temperature profile at all depths. Values of different variables predicted by a science and engineering model may have certain relationships (guided by physical laws) across space and time. For example, in the lake temperature monitoring application, predicted values of the temperature at different depths should be such that denser water is at lower depth (note that water is heaviest at 4 degree centigrade). As another example, changes in temperature profile across time involves transfer of energy and mass across

different layers of a lake that must be conserved according to physical laws.

Meanwhile, physics-based numerical simulations have become indispensable in civil engineering applications, such as seismic risk mitigation, irrigation management, structural design and analysis, and structural health monitoring. Civil engineers and scientists may now utilize sophisticated models for real-world applications, with ultra-realistic simulations involving millions of degrees of freedom, thanks to the advancement of high-performance computers. However, in the civil engineering sector, such simulations are too time-consuming to be incorporated fully into an iterative design process. They are often restricted to the final validation and certification stages, while most design processes rely on simpler models. Accelerating complex simulations is an important problem to address since it would make it easier to apply numerical tools throughout the design process. The development of numerical methods for rapid simulations would also enable novel model applications such as improving construction productivity, which has yet to be fully utilized due to model complexity. Uncertainty quantification is another critical example of analysis that might be feasible if simulation costs were lowered substantially. Indeed, the physical system environment, which is generally unknown, affects the values of interest monitored in numerical simulations. In some situations, these uncertainties significantly impact simulation results, necessitating estimating probability distributions for the quantities of interest to assure the product's dependability. Neither an ML-only nor a scientific knowledge-only method can be considered sufficient for complicated scientific and technical applications. Researchers are beginning to investigate the continuum between mechanistic and ML models, synergizing scientific knowledge and data.

There have been several reviews on ML civil engineering. However, limited studies have been conducted on physics-based ML and synthesizing a road map for guiding subsequent research to advance the proper use of physics-based ML in civil engineering applications. Furthermore, there are few works focused on the fundamental physics-based ML models in civil engineering. This study investigates a more profound connection of ML methods with physics models. Even though the notion of combining scientific principles with ML models has only recently gained traction [95], there has already been a significant amount of research done on the subject. Researchers focus on physics models, ML models, and application scenarios to solve their problems in civil engineering.

2.4.1 Integrating Physics-Based Modeling With Machine Learning: A Survey

The goal of this survey is to bring these exciting developments to the ML community, to make them aware of the progress that has been made, and the gaps and opportunities that exist for advancing research in this promising direction. the authorshope that this survey will also be valuable for scientists who are interested in exploring the use of ML to enhance modeling in their respective disciplines. Please note that work on

this topic has been referred to by other names, such as "physics-guided ML," "physics-informed ML," or "physics-aware AI," although it covers many scientific disciplines. In this survey, the authorsalso use the terms "physics-guided" or "physics," which should be more generally or interpreted as science or scientific knowledge.

The focus of this survey is on approaches that integrate mechanistic modeling with ML, using primarily ideas from physics and other scientific disciplines. This distinguishes the survey from other works that focus on more general knowledge integration into machine learning [2, 193] and other works covering physics integration into ML in specific domains (e.g., cyber-physical systems [147], chemistry [136]). This survey creates a novel taxonomy specific to science-based knowledge and covers a wide array of both methodologies and categories.

First-principles models are used extensively in a wide range of engineering and environmental applications. Even though these models are based on known physical laws, in most cases, they are necessarily approximations of reality due to incomplete knowledge of certain processes, which introduces bias. In addition, they often contain a large number of parameters whose values must be estimated with the help of limited observed data, degrading their performance further, especially due to heterogeneity in the underlying processes in both space and time. The limitations of physics-based models cut across discipline boundaries and are well known in the scientific community (e.g., see Gupta et al. [127] in the context of hydrology).

ML models have been shown to outperform physics-based models in many disciplines (e.g., materials science [99, 159, 198], applied physics [15, 116], aquatic sciences, atmospheric science, biomedical science, computational biology). A major reason for this success is that ML models (e.g., neural networks), given enough data, can find structure and patterns in problems where complexity prohibits the explicit programming of a system's exact physical nature. Given this ability to automatically extract complex relationships from data, ML models appear promising for scientific problems with physical processes that are not fully understood by researchers, but for which data of adequate quality and quantity is available. However, the black-box application of ML has met with limited success in scientific domains due to a number of reasons [92, 95]: (i) while state-of-the-art ML models are capable of capturing complex spatio-temporal relationships, they require far too much labeled data for training, which is rarely available in real application settings, (ii) ML models often produce scientifically inconsistent results; and (iii) ML models can only capture relationships in the available training data, and thus cannot generalize to out-of-sample scenarios (i.e., those not represented in the training data).

The key objective here is to combine elements of physics-based modeling with state-of-the-art ML models to leverage their complementary strengths. Such integrated physics-ML models are expected to better capture the dynamics of scientific systems and advance the understanding of underlying physical processes.

Early attempts for combining ML with physics-based modeling in several applications (e.g., modeling the lake phosphorus concentration [82] and lake temperature dynamics) have already demonstrated its potential for providing better prediction accuracy with a much smaller number of samples as well as generalizability in out-of-sample scenarios. In many physical systems, governing equations are known, but direct numerical solutions of PDEs using common methods, such as the Finite Elements Method or the Finite Difference Method [63], are prohibitively expensive. In such cases, traditional methods are not ideal or sometimes even possible. A common technique is to use an ML model as a surrogate for the solution to reduce computation time [52]. In particular, NN solvers can reduce the high computational demands of traditional numerical methods into a single forward-pass of a NN. Notably, solutions obtained via NNs are also naturally differentiable and have a closed analytic form that can be transferred to any subsequent calculations, a feature not found in more traditional solving methods [106]. Especially with the recent advancement of computational power, neural networks models have shown success in approximating solutions across different kinds of physics-based PDEs [100], including the difficult quantum many-body problem and many-electron Schrödinger equation [80]. As a step further, deep neural networks models have shown success in approximating solutions across high dimensional physics-based PDEs previously considered unsuitable for approximation by ML [79, 173]. However, slow convergence in training, limited applicability to many complex systems, and reduced accuracy due to unawareness of physical laws can prove problematic.

In many physical systems, governing equations are known, but direct numerical solutions of PDEs using common methods, such as the Finite Elements Method or the Finite Difference Method [63], are prohibitively expensive. In such cases, traditional methods are not ideal or sometimes even possible. A common technique is to use an ML model as a surrogate for the solution to reduce computation time [52]. In particular, NN solvers can reduce the high computational demands of traditional numerical methods into a single forward-pass of a NN. Notably, solutions obtained via NNs are also naturally differentiable and have a closed analytic form that can be transferred to any subsequent calculations, a feature not found in more traditional solving methods [106]. Especially with the recent advancement of computational power, neural networks models have shown success in approximating solutions across different kinds of physics-based PDEs [12], including the difficult quantum many-body problem and many-electron Schrodinger equation. As a step further, deep neural networks models have shown success in approximating solutions across high dimensional physics-based PDEs previously considered unsuitable for approximation by ML. However, slow convergence in training, limited applicability to many complex systems, and reduced accuracy due to unawareness of physical laws can prove problematic.

When the governing equations of a dynamical system are known explicitly, they allow for more robust forecasting, control, and the opportunity for analysis of system stability and bifurcations through increased

interpretability [158]. Furthermore, if the learned mathematical model accurately describes the processes governing the observed data, it therefore can generalize to data outside of the training domain. However, in many disciplines (e.g., neuroscience, cell biology, finance, epidemiology) dynamical systems have no formal analytic descriptions. Often in these cases, data is abundant, but the underlying governing equations remain elusive. In this section, the authorsdiscuss equation discovery systems that do not assume the structure of the desired equation , but rather explore a space a large space of possibly nonlinear mathematical terms.

Advances in ML for the discovery of these governing equations has become an active research area with rich potential to integrate principles from applied mathematics and physics with modern ML methods. Early works on the data-driven discovery of physical laws relied on heuristics and expert guidance and were focused on rediscovering known, non-differential, laws in different scientific disciplines from artificial data [72, 109]. This was later expanded to include real-world data and differential equations in ecological applications [56]. Recently, general and robust data-driven discovery of potentially unknown governing equations has been pioneered by [27], where they apply symbolic regression to differences between computed derivatives and analytic derivatives to determine underlying dynamical systems. More recently, works have used sparse regression built on a dictionary of functions and partial derivatives to construct governing equations [32]. Lagergren et al. [107] expand on this by using ANNs to construct the dictionary of functions. These sparse identification techniques are based on the principle of Occam's Razor, where the goal is that only a few equation terms be used to describe any given nonlinear system.

This survey focuses primarily on improving the modeling of engineering and environmental systems that are traditionally solved using physics-based modeling. However, the general ideas of physics-informed ML have wider applicability, and such research is already being pursued in many other contexts. For example, there are several interesting works in system control which often involves reinforcement learning techniques (e.g., combining model predictive control with Gaussian processes in robotics [11], informed priors for neuroscience modeling, physics-based reward functions in computational chemistry, and fluidic feedback control from a cylinder). Other examples include identifying features of interest in the output of computational simulations of physics-based models (e.g., high-impact weather predictions, segmentation of climate models, and tracking phenomena from climate model data).

2.4.2 Physics Informed Deep Learning

With the explosive growth of available data and computing resources, recent advances in machine learning and data analytics have yielded transformative results across diverse scientific disciplines, including image recognition [105], natural language processing [111], cognitive science [108], and genomics [5]. However, more often than not, in the course of analyzing complex physical, biological or engineering systems, the

cost of data acquisition is prohibitive, and the authorsare inevitably faced with the challenge of drawing conclusions and making decisions under partial information. In this small data regime, the vast majority of state-of-the art machine learning techniques (e.g., deep/convolutional/recurrent neural networks) are lacking robustness and fail to provide any guarantees of convergence.

At first sight, the task of training a deep learning algorithm to accurately identify a nonlinear map from a few - potentially very high-dimensional input and output data pairs seems at best naive. Coming to the rescue, for many cases pertaining to the modeling of physical and biological systems, there an exist a vast amount of prior knowledge that is currently not being utilized in modern machine learning practice. Let it be the principled physical laws that govern the time-dependent dynamics of a system, or some empirical validated rules or other domain expertise, this prior information can act as a regularization agent that constrains the space of admissible solutions to a manageable size (for e.g., in incompressible fluid dynamics problems by discarding any non-realistic low solutions that violate the conservation of mass principle). In return, encoding such structured information into a learning algorithm results in amplifying the information content of the data that the algorithm sees, enabling it to quickly steer itself towards the right solution and generalize well even when only a few training examples are available.

The first glimpses of promise for exploiting structured prior information to construct data-efficient and physics-informed learning machines have already been showcased in the recent studies of [139, 148, 149]. There, the authors employed Gaussian process regression [8] to devise functional representations that are tailored to a given linear operator, and were able to accurately infer solutions and provide uncertainty estimates for several prototype problems in mathematical physics. Extensions to nonlinear problems were proposed in subsequent studies by Raissi et. al. [151, 152] in the context of both inference and systems identification. Despite the flexibility and mathematical elegance of Gaussian processes in encoding prior information, the treatment of nonlinear problems introduces two important limitations. First, in [151, 152] the authors had to locally linearize any nonlinear terms in time, thus limiting the applicability of the proposed methods to discrete-time domains and compromising the accuracy of their predictions in strongly nonlinear regimes.

Secondly, the Bayesian nature of Gaussian process regression requires certain prior assumptions that may limit the representation capacity of the model and give rise to robustness/brittleness issues, especially for nonlinear problems[140]. The authors have introduced physics informed neural networks, a new class of universal function approximators that is capable of encoding any underlying physical laws that govern a given data-set, and can be described by partial differential equations.

In this work, the authors design data-driven algorithms for inferring solutions to general nonlinear partial differential equations, and constructing computationally efficient physics-informed surrogate models. The resulting methods showcase a series of promising results for a diverse collection of problems in computational

science, and open the path for endowing deep learning with the powerful capacity of mathematical physics to model the world around us. As deep learning technology is continuing to grow rapidly both in terms of methodological and algorithmic developments, the authorsbelieve that this is a timely contribution that can benefit practitioners across a wide range of scientific domains. Specific applications that can readily enjoy these benefits include, but are not limited to, data-driven forecasting of physical processes, model predictive control, multi-physics/multiscale modeling and simulation.

The proposed methods should not be viewed as replacements of classical numerical methods for solving partial differential equations (e.g., finite elements, spectral methods, etc.). Such methods have matured over the last 50 years and, in many cases, meet the robustness and computational efficiency standards required in practice. The message here is that classical methods such as the Runge-Kutta time-stepping schemes can coexist in harmony with deep neural networks, and offer invaluable intuition in constructing structured predictive algorithms. Moreover, the implementation simplicity of the latter greatly favors rapid development and testing of new ideas, potentially opening the path for a new era in data-driven scientific computing. This will be further highlighted in the second part of this paper, in which physics informed neural networks are put to the test of data-driven discovery of partial differential equations.

In terms of future work, one pressing question involves addressing the problem of quantifying the uncertainty associated with the neural network predictions. Although this important element was naturally addressed in previous work employing Gaussian processes [151], it is not captured by the proposed methodology in its present form and requires further investigation.

In the second part [150], the authors have introduced physics informed neural networks, a new class of universal function approximators that is capable of encoding any underlying physical laws that govern a given data-set, and can be described by partial differential equations. In this work, the authorsdesign data-driven algorithms for discovering dynamic models described by parametrized nonlinear partial differential equations. The inferred models allow us to construct computationally efficient and fully differentiable surrogates that can be subsequently used for different applications including predictive forecasting, control, and optimization.

Although a series of promising results were presented, the reader may agree that this two-part treatise creates more questions than it answers. In a broader context, and along the way of seeking further understanding of such tools, the authorsbelieve that this work advocates a fruitful synergy between machine learning and classical computational physics that has the potential to enrich both fields and lead to high-impact developments.

2.5 Physics-Guided Recurrent Neural Networks

The Physics-Guided Recurrent Neural Network models (PGRNN) is a general framework for modeling physical phenomena with potential applications for many disciplines. The PGRNN model has a number of novel aspects:

1. Many temporal processes in environmental/engineering systems involve complex long-term temporal dependencies that cannot be captured by a plain neural network or a simple temporal model such as a standard RNN. In contrast, in PGRNN the authors use advanced ML models such as LSTM, which use the internal memory structure to preserve long-term temporal dependencies and thus has the potential to capture complex physical pattern that last over several months or years.

2. The PGRNN can incorporate explicit physical laws such as energy conservation or mass conservation. This is done by introducing additional variables in the recurrent structure to keep track of physical states that can be used to check for consistency with physical laws. In addition, the authorsgeneralize the loss function to include a physics-based penalty. Thus, the overall training loss is $L =$ Supervised loss (Y_{pred}, Y_{true}) + Physics-based Penalty, where the first term on the right hand side represents the supervised training loss between the predicted outputs Y_{pred} and the observed outputs Y_{true} (e.g., RMSE in regression or cross-entropy in classification), and the second term represents the physical consistency-based penalty. In addition, to favoring physically consistent solutions, another major side benefit of including physics-based penalty in the loss function is that it can be applied even to instances for which output (observed) data is not available since the physics-based penalty can be computed as long as input (driver) data is available. Note that in absence of physics based penalty, training loss can be computed only on those time steps where observed output is available. Inclusion of physics based loss term allows a much more robust training, especially in situations, where observed output is available on only a small number of time steps.

3. Physics based/mechanistic models contain a lot of domain knowledge that goes well beyond what can be captured as constraints such conservation laws. To leverage this knowledge, the authors generate a large amount of "synthetic" observation data by executing physics based models for a variety input drivers (that are easily available) and use the synthetic observation to pre-train the ML model. The idea here is that training from synthetic data generated by imperfect physical models may allow the ML model to get close enough to the target solution, so only a small amount of observed data (ground truth labels) is needed to further refine the model. In addition, the synthetic data is guaranteed to be physically consistent due to the nature of the process model being founded on physical principles.

The PGRNN is developed for the purpose of predicting lake water temperatures at various depths at the daily scale. The temperature of water in a lake is known to be an ecological "master factor" that controls the

growth, survival, and reproduction of fish (Roberts et al. 2013). Warming water temperatures can increase the occurrence of aquatic invasive species, which may displace fish and native aquatic organisms, result in more harmful algal blooms. Understanding temperature change and the resulting biotic "winners and losers" is timely science that can also be directly applied to inform priority action for natural resources. Given the importance of this problem, the aquatic science community has developed numerous models for the simulation of temperature, including the General Lake Model (GLM), which simulates the physical processes (e.g., vertical mixing, and the warming or cooling of water via energy lost or gained from fluxes such as solar radiation and evaporation, etc.). As is typical for any such model, GLM is only an approximation of the physical reality, and has a number of parameters (e.g., water clarity, mixing efficiency, and wind sheltering) that often need to be calibrated using observations.

2.5.1 Hydronets: Leveraging River Structure for Hydrologic Modeling

HydroNets [131] is a family of deep neural network models designed for hydrologic forecasting. HydroNets leverages the prior knowledge of the sub-basins' structure of a hydrologic region. HydroNets also enforce some weight sharing between sub-basins, resulting in a shared model and basin-specific models that correspond to the general-physical hydrologic modeling which is shared among basins vs. the basin-specific modeling that account for basin properties. The proposed architecture is modular, thus making it convenient to understand and improve. the authorspresent experimental results over two regions in India which convincingly show that the proposed model utilizes learning examples from the whole region, avoids overfitting, and performs better when training data is scarce.

2.5.2 Graph-based Reinforcement Learning for Active Learning in Real Time

The last few years have witnessed a surge of interest in building ML methods for scientific applications in diverse disciplines, e.g., hydrology [91], biological sciences [214], and climate science [6]. Given the promising results from previous research, expectations are rising for using ML to accelerate scientific discovery and help address some of the biggest challenges that are facing humanity such as water quality, climate, and healthcare. However, ML models focus on mining the statistical relationships from data and thus often require large amount of labeled observation data to tune their model parameters.

Collecting labeled data is often expensive in scientific applications due to the substantial manual labor and material cost required to deploy sensors or other measuring instruments. For example, collecting water temperature data commonly requires highly trained scientists to travel to sampling locations and deploy sensors within a lake or stream, incurring personnel and equipment costs for data that may not improve model predictions. To make the data collection more efficient, the authorsaim to develop a data-driven method that

assists domain experts in determining when and where to deploy measuring instruments in real time so that data collection can be optimized for training ML models.

Active learning has shown great promise for selecting representative samples [61, 164]. In particular, it aims to find a query strategy based on which the authorscan annotate samples that optimize the training of a predictive ML model. Traditional pool-based active learning methods are focused on selecting query samples from a fixed set of data points. These techniques have been widely explored in image recognition [68, 116] and natural language processing [169, 221]. These approaches mostly select samples based on their uncertainty level, which can be measured by Bayesian inference [13] and Monte Carlo drop-out approximation [67]. The samples with higher uncertainty tend to stay closer to the current decision boundary and thus can bring higher information gain to refine the boundary. Some other approaches also explore the diversity of samples so that they can annotate samples different with those that have already been labeled [35, 168, 201].

However, the pool-based active learning approaches cannot be used in scientific problems as they assume all the data points are available in a fixed set while scientific data have to be annotated in real time. When monitoring scientific systems, new labels can only be collected by deploying sensors or other measuring instruments. Such labeling decisions have to be made immediately after the authorsobserve the data at the current time, which requires balancing the information gain against the budget cost. Hence, these labeling decisions are made without access to future data and also cannot be changed afterwards.

Moreover, existing approaches do not take into account the representativeness of the selected samples given their spatial and temporal context. Scientific systems commonly involve multiple physical processes that evolve over time and also interact with each other. For example, in a river network, different river segments can have different thermodynamic patterns due to different catchment characteristics as well as climate conditions. Connected river segments can also interact with each other through the water advected from upstream to downstream segments. In this case, new annotated samples can be less helpful if they are selected from river segments that have similar spatio-temporal patterns with previously labeled samples. Instead, the model should take samples that cover different time periods and river segments with distinct properties.

To address these challenges, the authors propose a new framework Graph-based Reinforcement Learning for Real-Time Labeling (GR-REAL), in which the authors formulate real-time active learning problem as a Markov decision process. This proposed framework is developed in the context of modeling streamflow and water temperature in river networks but the framework can be generally applied to many complex physical systems with interacting processes. The method makes labeling decisions based on the spatial and temporal context of each river segment as well as the uncertainty level at the current time. Once the authors determine the actions of whether to label each segment at the current time step, the collected labels can be used to refine

the way the authors represent the spatio-temporal context and estimate uncertainty for the observed samples at the next time step (i.e., the next state).

In particular, the proposed framework consists of a predictive model and a decision model. The predictive model extracts spatial and temporal dependencies from data and embeds such contextual information in a state vector. The predictive model also generates final predictions and estimates uncertainty based on the obtained embeddings. At the same time, the decision model is responsible for determining whether the authors will take labeling actions at the current time step based on the embeddings and outputs obtained from the predictive model. The collected labels are then used to refine the predictive model. the authors train the decision model via reinforcement learning using past observation data. During the training phase, the reward of labeling each river segment at each time step can be estimated as the expectation of accumulated performance improvement via dynamic programming over training sequential data.

Since this proposed data-driven method requires separate training data from the past history, which can be scarce in many scientific systems, the authors also propose a way to transfer knowledge from existing physics-based models which are commonly used by domain scientists to study environmental and engineering problems. The transferred knowledge can be used to initialize the decision model and thus less training data is required to fine-tune it to a quality model. the authors evaluate the proposed framework in predicting streamflow and water temperature in the Delaware River Basin. The proposed method produces superior prediction performance given limited budget for labeling. the authors also show that the distribution of collected samples is consistent with the dynamic patterns in river networks.

The authors evaluate the proposed framework in predicting streamflow and water temperature in the Delaware River Basin. The proposed method produces superior prediction performance given limited budget for labeling. the authors also show that the distribution of collected samples is consistent with the dynamic patterns in river networks.

In this paper, the authors propose the GR-REAL framework which uses the spatial and temporal contextual information to select query samples in real time. the authors demonstrate the effectiveness of GR-REAL in selecting informative samples for modeling streamflow and water temperature in the Delaware River Basin. the authors also show that policy transfer can further improve the performance when the authors have less training data. The proposed method may also be used to measure other water quality parameters for which sensors are costly or too difficult to maintain (e.g., metal, nutrients, or algal biomass).

While GR-REAL achieves better predictive performance, it estimates potential reward based on accuracy improvement and thus remains limited in selecting samples that indeed help understand an ecosystem. For example, the GR-REAL ignores low-flow segments as they contribute less to the overall accuracy loss. Studying this element of GR-REAL has promise for future work.

2.5.3 Physics-Guided Recurrent Graph Networks

Recent works have shown the promise of integrating physics into ML models in improving the predictive performance and generalizability in scientific problems. This is commonly conducted in several ways, including physics-guided model architectures [10, 134], physics-guided loss functions [91, 156], and other hybrid approaches [194].

There are several ways to incorporate known physics into ML models. For example, one can embed known physical principles into neural networks by ascribing physical meaning for certain neurons in the NN. For example, [133] build a new architecture to insert physics-constrained variables as the intermediate variables in the convolutional neural networks. This method has been tested for predicting drag force on particle suspensions in moving fluids and has achieved improved performance. Another direction is to use machine learning architecture to encode invariance and symmetries that are inherent of a physical system. In turbulence modeling and fluid dynamics [119] defines a tensor basis neural network to embed the fundamental principle of rotational invariance into neural networks for improved prediction accuracy.

When applied to systems with interacting processes, e.g., a river network, the authors need to build ML models that can handle such interactions. The Graph Convolutional Networks (GCN) model has proven to be effective in automatically modeling node interactions in a graph. The use of GCN has also shown improved prediction accuracy in several scientific problems [146, 209, 222]. In a previous work, Jia et al. [92] has leveraged physics to guide the extraction of hidden variables that are propagated in the GCN model. This method has been shown to produce better prediction accuracy as well as improved generalizability.

The architecture of Physics-Guided Recurrent Graph Networks (PGRGrN) is based on RNN and GCN, which explicitly captures the spatial interactions among different river segments as well as their temporal dynamics. Modeling of the spatial and temporal context is critical for the global ML model as it enables learning of different behavior patterns for different river segments even when they have similar input features on certain dates.

This architecture proposes to utilize the intermediate variables simulated by the physics-based model to guide the learning process of the graph neural networks. This approach aims to enforce a physical interpretation to latent variables learned from each river segment by transferring the prior knowledge encoded by the physics-based model to the proposed ML model. This architecture design a new loss function to ensure that the global ML model can simultaneously capture the local patterns of all the different segments. The local patterns of each segment can be extracted using an individual ML model trained only for this segment using simulation data (which is plentiful). Then during the training of the global ML model, the authors use a distance-based loss function, the contrastive loss function, to enforce its consistency with the extracted local

patterns.

Data assimilation has been widely used in physics-based modeling approaches with the aim to optimally adjust the model state of a system with recent observations. There are many data assimilation approaches used for physics-based models and they differ in their assumptions about the distributions of the model and data predictions and speed of computation. The Kalman filter, which iteratively estimates the next state through a forward process and then updates the state using new observations, is a popular data assimilation technique for physics-based models. Many variations of the Kalman filter have also been implemented [59]. Despite their extensive use, these approaches can be computationally expensive to implement when the authors have a large state space and/or non-linear system dynamics. Recently, researchers have started to use neural networks as an alternative way for implementing data assimilation [30, 60]. Note that this is different from online incremental learning in that model parameters remain the same but only states are changed. The intuition of data assimilation is to adjust the model state when they are disturbed by external factors that are not captured by input features. For example, Brajard et al. [30] propose a two-stage process: (1) in the training phase, model parameters (i.e., network weights) are trained using observations, and (2) in the data assimilation phase, model parameters are fixed and the model state is optimized to match observations using back-propagation.

Neural network models require an initial choice of model parameters before training. Poor initialization can cause models to anchor in local minima, which is especially true for complex models. Transfer learning has been widely used in computer vision [180], where the pretrained models from a related large-scale dataset are fine-tuned with limited training data to fit the target task. In scientific problems, the authors can use a physics-based model's simulated data to pre-train the ML model. This approach can also alleviate data paucity issues. Read et al. [156] show that recurrent neural networks after being pre-trained using simulation outputs are able to generalize better to unseen scenarios than pure physics-based models or machine learning models. Jia et al. also extensively discuss this strategy [92]. They pre-train their Physics-Guided Recurrent Neural Network models for lake temperature modeling on simulated data generated from a physics-based model and fine-tune it with little observed data. They show that pretraining can significantly reduce the training data needed for a quality model. Moreover, they show that machine learning models can still benefit from such pre-training process by learning general, physically consistent patterns (e.g., seasonal cycles) even when physical simulations are biased.

2.5.4 Physics-Guided Machine Learning from Simulation Data

In [92], the authors propose a new framework, simulation-guided learning (SIMLR), which extracts the general physical knowledge jointly from multiple sets of physical simulations with imperfect parameterizations.

The authors also explore the relationship between observation data and simulation data and identify parameter settings that produce the most accurate predictions over different locations and time periods. In particular, the authors first build a spatial-temporal network (STN) architecture to represent the spatial and temporal relationships in the dynamical system. Given that most physical parameters determine specific conditions that control how the system states react to external changes, the authors represent such conditioning factors using a set of gating variables in the ML architecture. The gating variables are used to filter the information from the current time step, previous time steps, and the spatial neighborhood. The filtered information is combined to update the state of the ML model. Then the authors propose a new pre-training strategy that leverages general physical patterns from different sets of simulation data to inform the initialization of the STN model. The idea is that this initialized model can be easily adjusted to fit each set of simulation data by slightly altering gating variables. After the initialization, the authors further refine the model using true observations via a contrastive learning process. The contrastive learning process aims to explore the similarity of relations between observations and different sets of simulation data and further transfer the knowledge from specific simulations that are closer to the observed reality.

The authors evaluate the performance in two societally relevant applications, modeling water temperature in a lake system and water temperature in river networks. Although predicting the same variable, these two applications have distinct spatio-temporal drivers of water temperature and focal parameters for physics-based calibration. The authors demonstrate the effectiveness of model initialization using general physical knowledge and show that the method can achieve good predictive performance even with very sparse observation data. The authors also analyze the similarity relationships learned from the contrastive loss and provide scientific interpretability. The method has shown promise in discovering variations of physical parameters across space and time while traditional physics-based model can often take fixed parameter values. Moreover, the method under the guidance of general physical relationships can better generalize to different scenarios.

2.6 Turbulent Flows Reconstruction

Computational fluid dynamics (CFD) has proven to be a very effective research tool in a very wide variety of disciplines, including engineering, science, medicine and more [199]. For its applications in turbulent flows, however, the range of the temporal & spatial scales is too broad to be captured by brute force direct numerical simulations (DNS) [49]. Large eddy simulation (LES) provides an alternative, by filtering the small-scale scales of transport and concentrating on the larger scale energy containing eddies [161]. By this filtering, LES can be conducted on coarser grids as compared to those required by DNS. The penalty, understandably, is that LES generated data are of lower accuracy compared to DNS. Appraisal of LES predictions and assessments

of its fidelity as compared to DNS, have been of interest in the turbulence research community for the past several decades [74, 144].

Machine learning, including super-Resolution methods [141], have shown great success in reconstructing high-resolution data in a variety of commercial applications. For example, convolutional neural networks (CNNs) and their extensions, e.g., SRCNN [53], RCAN [219], and SRGAN [113], have proven very effective in directly mapping low-resolution images to high-resolution images. The effectiveness of these methods mainly come from the power of CNNs in automatically extracting representative spatial features through deep layers. An alternative solution is to consider super-resolution as an inverse modeling problem [71, 126] with the constraints that the down-sampled version of the underlying high-resolution data should be consistent to the observed low-resolution data.

2.6.1 Reconstructing High-Resolution Turbulent Flows using Physics-Guided Neural Networks

Super-resolution techniques are starting to be used in turbulence research [65, 120, 210]. However, there are several major challenges that must be overcome, before they can be employed for routine applications. First, turbulent flow data often exhibit significant variability. In the absence of underlying physical processes, machine learning models are prone to learning spurious patterns that fit statistical characteristics of available training data collected from a specific time period, but cannot generalize to other time intervals. This can be further exacerbated by limited training data. Second, existing super-resolution algorithms can have degraded performance in CFD because of the huge information loss caused by the large resolution gap. For example, LES data can be of more than 8 times lower resolution compared to DNS data along each axis. Hence, standard statistical interpolation methods may fail to capture fine-level flow dynamics resulting from underlying physical relationships and constraints. Third, existing machine learning models are not designed to deal with the discrepancy between different simulation strategies (i.e. LES, DNS and/or others). In general, the available simulations at a coarser resolution are not simply a down-sampled version of high-resolution simulations.

In this paper, the authors develop a new method, termed Physics-Guided Super-Resolution Network (PGSRN), to improve the reconstruction of high-resolution turbulent flow data. This development is by leveraging known physical constraints and explicitly exploring the discrepancy & the consistency between different simulations. First, the authors generalize the loss function of the super-resolution model by incorporating the divergence-free velocity-field constraint as required in incompressible flows. Second, the authors introduce a hierarchical generative architecture by decomposing the data reconstruction into two steps: (i) transform low-resolution flow data into a down-sampled version of high-resolution data, and (ii) reconstruct high-resolution flow data from the down-sampled version. Step (i) allows explicitly modeling the data dis-

crepancy due to different simulation methods used to generate low-resolution and high-resolution data. Step (ii) is to recover the fine-level details of flow data. Finally, the authors introduce a degradation process to further regularize reconstructed data by imposing the consistency between different simulations. Here the authors represent the degradation process by a forward model (by framing super-resolution as an inverse problem) that maps high-resolution data to low-resolution data. The forward model output of reconstructed data can then be compared against low-resolution simulations for consistency assessment. the authors further extend the degradation process as a feature extractor and introduce an adversarial loss on the extracted features from high-resolution data, which helps improve the modeling of fine-level fluid dynamics.

For the purpose of demonstration, the authors consider a variant of the Taylor-Green vortex (TGV) [29]. This is a three-dimensional incompressible flow and is simulated within a box with periodic boundary conditions. The TGV provides a suitable setting for the demonstration as it exhibits several salient features of turbulent transport. In this flow, the original vortex collapses into turbulent worm-like structures which become progressively more turbulent until viscosity eventually dissipates the large scale vortical structures. The authors compare the proposed method against several existing super-resolution algorithms to reconstruct DNS data of TGV. The authors also demonstrate the effectiveness of each component in the proposed method by showing the improvement both qualitatively and quantitatively.

Although the method has been developed in the context of simulating fluid dynamics, the involved techniques can be widely used for other important scientific problems. For example, simulations of cloud-resolving models (CRM) at sub-kilometre horizontal resolution are critical for effectively representing boundary-layer eddies and low clouds. However, it is not feasible to generate simulations at such fine resolution even with the most powerful commuters expected to be available in the near future. Hence, the method developed in this paper can provide great potential for reconstructing high-resolution simulations.

Additionally, as shown in the cross-time experiment, the method remains limited in reconstructing long-term data. This requires new mechanisms to enforce underlying physical processes on fluid dynamics (e.g., Navier-Stokes equation). Furthermore, the authors plan to introduce other related parameters besides velocity (e.g., mass and pressure) to supplement and further optimize the model. Last, the authors also will introduce other domain's metrics (e.g., Reynolds Stress and Kinetic Energy [190]) as the training loss to enhance the trustworthiness of the model when it is deployed for long-term and large-scale simulations.

2.6.2 Deep Learning Methods for Super-Resolution Reconstruction of Turbulent Flows

In [120], two deep learning models, i.e., the SCNN and MTPC, are developed for the super-resolution reconstruction of turbulent flows. They all take low-resolution flow information as an input. However, the SCNN takes one instantaneous snapshot as an input, while the MTPC takes a temporal sequence of snapshots as an

input. Therefore, the MTPC has the advantage of drawing extra temporal information from adjacent frames.

To see whether the deep learning methods are able to reproduce turbulence, two canonical turbulent problems were tested. For the isotropic turbulence, the reproduced energy spectra and predicted PDF of the normalized velocity gradients by the MTPC are close to those of the DNS result. For the turbulent channel flow, the deep-learning-based approaches greatly enhanced the spatial resolution in different wall regions and layers. The correlation coefficients between reconstructed data and the reference data are high in the outer layer and log-law region where turbulence dominates. However, the coefficients are not so high in the viscosity-dominated region where there are the most vigorous turbulent activities. All assessments in both cases show that the MTPC greatly improved the quality of the LR input and outperformed the SCNN and bicubic interpolation method, especially at small scales. The extra temporal information from consecutive snapshots helps the MTPC to generate more physically reasonable results. On the other hand, because there are more input snapshots to handle, the MTPC spends more time on prediction compared with static models. All the experiments were performed with 2D snapshots, but the networks used in this study can be easily extended to 3D cases by using 3D convolution kernels.

Although deep-learning methods achieve great progress, there are still some challenges. First, the DL models are not able to exactly reproduce the kinetic energy several orders of magnitude smaller than the total energy, i.e., the very small spatial structures. Second, the DL methods show performance diversity in different directions in the anisotropic turbulence case even though a special normalization method is used to convert data in the three directions into the same order of magnitude. These problems may be eliminated by introducing novel machine learning methods like unsupervised learning or including prior physical knowledge in the training process. That may be part of the future work.

The success of the DL models in reconstructing subgrid flow variables probably inspires the development of subgrid models in the CFD. The SR technology can also serve as the post-processing tool to denoise, correct, or enrich data from experiments and numerical simulations. the authors believe that the super-resolution technology would have a board practical application in the fluid dynamics with the increasing volumes of data accessible.

2.6.3 Learning a Deep Convolutional Network for Image Super-Resolution

The sparse-coding-based method [212, 213] is one of the representative methods for external example-based image super-resolution. This method involves several steps in its pipeline. First, overlapping patches are densely extracted from the image and pre-processed (e.g., subtracting mean). These patches are then encoded by a low-resolution dictionary. The sparse coefficients are passed into a high-resolution dictionary for reconstructing high-resolution patches. The overlapping reconstructed patches are aggregated (or averaged) to

produce the output. Previous SR methods pay particular attention to learning and optimizing the dictionaries [212, 213] or alternative ways of modeling them [24, 37]. However, the rest of the steps in the pipeline have been rarely optimized or considered in an united optimization framework.

In this paper, the authors show the aforementioned pipeline is equivalent to a deep convolutional neural network [112]. Motivated by this fact, the authors directly consider a convolutional neural network which is an end-to-end mapping between low- and high-resolution images. The proposed method differs fundamentally from existing external example-based approaches, in that the proposed method does not explicitly learn the dictionaries [212, 213] or manifolds [24, 37] for modeling the patch space. These are implicitly achieved via hidden layers. Furthermore, the patch extraction and aggregation are also formulated as convolutional layers, so are involved in the optimization. In the method, the entire SR pipeline is fully obtained through learning, with little pre/post-processing.

The authors name the proposed model Super-Resolution Convolutional Neural Network (SRCNN). The proposed SRCNN has several appealing properties. First, its structure is intentionally designed with simplicity in mind, and yet provides superior accuracy comparing with state-of-the-art example-based methods. Second, with moderate numbers of filters and layers, the method achieves fast speed for practical on-line usage even on a CPU. The method is faster than a series of example-based methods, because it is fully feed-forward and does not need to solve any optimization problem on usage. Third, experiments show that the restoration quality of the network can be further improved when (i) larger datasets are available, and/or (ii) a larger model is used. On the contrary, larger datasets/models can present challenges for existing example-based methods. Overall, the contributions of this work are mainly in three aspects: 1) The authors present a convolutional neural network for image super-resolution. The network directly learns an end-to-end mapping between low- and high-resolution. 2) The authors establish a relationship between the deep-learning-based SR method and the traditional sparse-coding-based SR methods. This relationship provides a guidance for the design of the network structure. 3) The authors demonstrate that deep learning is useful in the classical computer vision problem of super-resolution, and can achieve good quality and speed.

In conclusion. the authors have presented a novel deep learning approach for single image super-resolution (SR). Conventional sparse-coding-based image super-resolution methods can be reformulated into a deep convolutional neural network. The proposed approach, SRCNN, learns an end-to-end mapping between low- and high-resolution images, with little extra pre/post-processing beyond the optimization. With a lightweight structure, the SRCNN has achieved superior performance than the state-of-the-art methods. Additional performance can be further gained by exploring more hidden layers/filters in the network, and different training strategies. Besides, the proposed structure, with its advantages of simplicity and robustness, could be applied to other low-level vision problems, such as image deblurring or simultaneous SR + denoising. One could also

investigate a network to cope with different upscaling factors.

2.6.4 tempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow

Generative models were highly successful in the last years to represent and synthesize complex natural images [75]. These works demonstrated that deep CNNs are able to capture the distribution of, e.g., photos of human faces, and generate novel, previously unseen versions that are virtually indistinguishable from the original inputs. Likewise, similar algorithms were shown to be extremely successful at generating natural high-resolution images from a coarse input [97]. However, in their original form, these generative models do not take into account the temporal evolution of the data, which is crucial for realistic physical systems. In the following, the authors will extend these methods to generate high-resolution volumetric data sets of passively advected flow quantities, and ensuring temporal coherence is one of the core aspects that the authors will focus on below. The authors will demonstrate that it is especially important to make the training process aware of the underlying transport phenomena, such that the network can learn to generate stable and highly detailed solutions.

Capturing the intricate details of turbulent flows has been a longstanding challenge for numerical simulations. Resolving such details with discretized models induces enormous computational costs and quickly becomes infeasible for flows on human space and time scales. While algorithms to increase the apparent resolution of simulations can alleviate this problem [101], they are typically based on procedural models that are only loosely inspired by the underlying physics. In contrast to all previous methods, the algorithm represents a physically-based interpolation, that does not require any form of additional temporal data or quantities tracked over time. The super-resolution process is instantaneous, based on volumetric data from a single frame of a fluid simulation. The authors found that inference of high-resolution data in a fluid flow setting benefits from the availability of information about the flow. In the case, this takes the shape of additional physical variables such as velocity and vorticity as inputs, which in turn yield means for artistic control. A particular challenge in the field of super-resolution flow is how to evaluate the quality of the generated output. As the authors are typically targeting turbulent motions, a single coarse approximation can be associated with a large variety of significantly different high-resolution versions. As long as the output matches the correlated spatial and temporal distributions of the reference data, it represents a correct solution. To encode this requirement in the training process of a neural network, the authors employ so-called generative adversarial networks (GANs). These methods train a generator, as well as a second network, the discriminator that learns to judge how closely the generated output matches the ground truth data. In this way, the authors train a specialized, data-driven loss function alongside the generative network, while making sure it is differentiable and compatible with the training process. the authors not only employ this adversarial approach for the smoke

density outputs, but the authors also train a specialized and novel adversarial loss function that learns to judge the temporal coherence of the outputs.

The authors additionally present practices to set up a training pipeline for physics-based GANs. E.g., the authors found it particularly useful to have physics-aware data augmentation functionality in place. The large amounts of space-time data that arise in the context of many physics problems quickly bring typical hardware environments to their limits. As such, the authors found data augmentation crucial to avoid overfitting. They also explored a variety of different variants for setting up the networks as well as training them, and the authors will evaluate them in terms of their capabilities to learn high-resolution physics functions below. To summarize, the main contributions of the work are: 1) a novel temporal discriminator, to generate consistent and highly detailed results over time, 2) artistic control of the outputs, in the form of additional loss terms and an intentional entangling of the physical quantities used as inputs, 3) a physics aware data augmentation method, 4) and a thorough evaluation of adversarial training processes for physics functions.

A first conditional GAN approach for four-dimensional data sets, and it is possible to train generators that preserve temporal coherence using the novel time discriminator. The network architecture of this temporal discriminator, which ensures that the generator receives gradient information even for complex transport processes, makes it possible to robustly train networks for temporal evolutions. This discriminator improves the generation of stable details as well as the learning process itself. At the same time, the fully convolutional networks can be applied to inputs of arbitrary size, and the approach provides basic means for art direction of the generated outputs. It is very promising to see that the CNNs are able to benefit from coherent, physical information even in complex 3D settings, which led to reduced network sizes. Overall, the authors believe that the contributions yield a robust and very general method for generative models of physics problems, and for super-resolution flows in particular.

CHAPTER 3

A New Hybrid Automaton Framework with Partial Differential Equation Dynamics

3.1 Cyber-Physical Systems and Hybrid Automata

CPSs have the potential to revolutionize the development of technology worldwide, and in recent years, numerous researchers and companies have invested significant time and money into ensuring that these systems will realize their economic and societal potential [114]. However, since CPS fundamentally integrate computational and physical processes, there are unique challenges in designing and analyzing these systems. In fact, the functional correctness of CPS relies deeply on the dynamics of their physical environment and the discrete control decisions of their computational units [104]. Within hybrid systems, the framework of HA has demonstrated considerable utility in capturing the complex interaction between the discrete and continuous parts of a CPS. Additionally, it allows for a formal analysis of the safety and reliability of CPS [104]. However, classically, this modeling framework has catered to systems with dynamics that can be described by ODEs, and hardly any attention has been paid to systems with dynamics described by PDEs [189].

This chapter presents the syntax, semantics, and decidability results of a new type of HA with partial differential equation dynamic, PDHA, whose continuous dynamics are described by partial differential equations. While classically the dynamics of HA are described by ODEs and differential inclusions, PDHA are capable of describing the behaviour of CPS with continuous dynamics that cannot be modelled using the canonical hybrid systems framework. For the purposes of analysing PDHA, we propose another model called the DSPDHA which handles discrete spatial domains. Additionally, we formally outline concepts such as time trajectories, PDHA and DSPDHA execution, zeno behaviour, among others to make thorough analysis convenient. We conclude with two illustrative examples in order to exhibit the nature of PDHA and DSPDHA.

3.2 Partial Differential Equations

A PDE is an equation involving *partial derivatives*. This is not so informative so let's break it down.

3.2.1 Differential Equations

An ODE is an equation for a function that depends on one independent variable, involves the independent variable, the function, and derivatives of the function:

$$F(t,u(t),u^{(1)}(t),u^{(2)}(t),u^{(3)}(t),...,u^{(m)}(t))=0. \tag{3.1}$$

Equation (3.1) is an example of an ODE of degree m, where m is the highest order of the derivative in the equation. Solving an equation like this on an interval $t \in [0,T]$ equates to finding a function $t \to u(t) \in \mathbb{R}$ with the property that u and its derivatives couple in such a way that this equation is true for all values of $t \in [0,T]$. The problem can be expanded to vector-valued functions by replacing the real-valued u by a vector-valued one $u(t) = (u_1(t), u_2(t), ..., u_N(t))$. In such a case we usually talk about systems of ODEs.

3.2.2 Partial Derivative

When you have function that depends upon several variables, you can differentiate with respect to each variable while holding the other variables constant. This is the basic idea behind *partial derivatives*. As an example, consider a function that depends on two real variables:

$$u : \mathbb{R}^n \to \mathbb{R}. \tag{3.2}$$

For $n = 2$, we sometimes visualize such a function by considering its graph as a surface in $\mathbb{R}^3$ given by the collection of points:

$$(x,y,z) \in \mathbb{R}^3 : z = u(x,y). \tag{3.3}$$

We can compute the derivative with respect to x while holding y fixed. This yields u_x which can also be expressed as $\partial_x u$, $\frac{\partial u}{\partial x}$, and $\frac{\partial}{\partial x}$. Similarly, we can hold x fixed and differentiate with respect to y. A partial differential equation is an equation for a function that depends on more than one independent variable, involves the independent variables, the function, and the partial derivatives of the function:

$$\begin{aligned} &F(x,y,u(x,y),u_x(x,y),u_y(x,y),u_{xx}(x,y),u_{xy}(x,y),u_{yx}(x,y),\\ &u_{yy}(x,y))=0. \end{aligned} \tag{3.4}$$

Equation (3.4) is an example of a second degree PDE. Solving an equation such as this corresponds to finding

Types of PDEs	Coefficient condition	Examples
parabolic equation	$B^2 - AC = 0$	heat equation: $u_t - \alpha\Delta u = f$
hyperbolic equation	$B^2 - AC > 0$	wave equation: $u_{tt} - c^2 u_{xx} = 0$
elliptic equation	$B^2 - AC < 0$	Poisson's equation: $\Delta u = f$

Table 3.1: The definition of the types of PDEs, their coefficient conditions, and examples.

a function $(x,y) \to u(x,y)$ with the property that u and its partial derivatives satisfy equation (3.4). In a similar fashion to the case of ODEs, we can expand equation (3.4) to describe vector valued functions by replacing the real-valued function u with a vector-valued one $\mathbf{u}(t) = (u_1(t), u_2(t), ..., u_N(t))$. In this framework, we usually talk about a system of PDEs.We can also write (3.4) in its explicit form:

$$\begin{aligned} &Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu + G = 0, \\ &u_{xx} = \frac{\partial^2 u}{\partial x^2}, u_{xy} = \frac{\partial^2 u}{\partial x \partial y}, u_{yy} = \frac{\partial^2 u}{\partial y^2}, u_x = \frac{\partial u}{\partial x}, u_y = \frac{\partial u}{\partial y}, \end{aligned} \tag{3.5}$$

where A, B, C, D, E, F, G are constants.

As is common in the PDE literature, we write u_t to represent first order derivative of time, and u_x and u_{xx} to represent the first and second order derivative with respect to space. We refer to ∇u as the *gradient* of u and Δu as the *Laplacian* of u. Additionally, some linear, second-order PDEs, can be classified as *parabolic*, *hyperbolic*, or *elliptic*. Table 3.1 captures the relation between the coefficients in equation (3.5) and the equation type, providing a guide to specify appropriate initial and boundary conditions and the smoothness of the solutions.

It is worth noting that in PDE modeling, there is a *domain* of discourse over which the PDE is valid for capturing physical phenomena. From physical intuition, it is necessary to specify boundary conditions in order for the solution to be determined. Table 3.2 summarizes several useful classes of boundary conditions (BCs). For the *Dirichlet* boundary condition, the value is specified on Γ. The *Neumann* boundary condition specifies the derivative on Γ and the *Robin* boundary condition provides mixed condition.

3.3 Running Examples

To illustrate the main notions of our framework, we use two running examples throughout the chapter. The components of these running examples are depicted in Figures 3.1 and 3.2. The heater model consists of three devices: (i) a rod that can be heated using a gas burners from anywhere, (ii) a long gas burner that can be turned on or off partially in any region, and (iii) a thermometer that monitors the temperature of the rod

Types of BCs	Values on boundary Γ	Examples
Dirichlet	$u(x) = g(x)$	fixed temperature is given on Γ for heat equation
Neumann	$\frac{\partial u}{\partial n} = g(x)$	wave is reflected on Γ for wave equation
Robin	$\frac{\partial u}{\partial x} + \beta(x)u = g(x)$	heat exchange is valid on Γ for heat equation

Table 3.2: Types of Boundary Conditions, their values on boundary and examples.

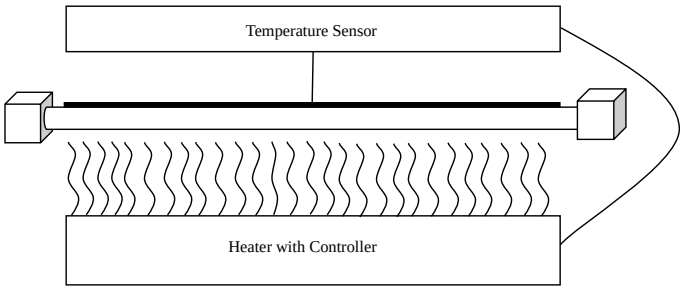


Figure 3.1: Diagram for heater model.

at every location and periodically issues signals when the temperature of the rod is above or below certain a threshold. The goal is to design a control policy that maintains ensures that the temperature of the rod remains within a given range.

The second model we consider is a traffic model that was originally proposed in [44]. The system consists of a highway segment in which vehicles flow in an out. Vehicles maintain a high speed when the traffic flow density is below a given critical density and when the density exceeds the the critical threshold, the speed of the traffic wave and the overall flow capacity decrease.

3.3.1 Heater Model

We first describe the behavior of the temperature of the rod. When the gas burner is OFF, the temperature of the rod, denoted by the variable u, decreases according to the following differential equation: $u(x,t)_t - \alpha u(x,t)_{xx} = 0$ where α is the thermal diffusivity. The thermal diffusivity measures the rate of heat transfer of a material from the hot side to the cold side. The boundary conditions are $u(0,t) = 0$ and $u(L,t) = 0$, which denote that at location $x = 0$ and $x = L$ the temperature is fixed to be 0, and heat is absorbed. X denotes the space coordinate and t denotes time. This law however, is only true when the temperature of the corresponding position on the rod is greater than 0.7. When the heater is OFF, and the temperature of a position on the rod is below 0.4, the heater at that position will be turned on. The heater function is defined

Figure 3.2: Diagram for traffic model.

as $f(x) = -\frac{x}{L} + 1$. The maximum of the heater function appears at $x = 0$ and the heating effect is 0 at $x = L$. During the the heater ON mode, the temperature is governed by the equation $u(x,t)_t - \alpha u(x,t)_{xx} = f(x)$ where $f(x)$ is is the function defined above. As one can see from the description of the evolution of the temperature, the system is not purely continuous. The system can switch from one mode to another by turning the heater on and off at a specific location on the rod.

3.3.2 Traffic Flow Model

The traffic model describes how traffic conditions may behave on a given highway. In free flow mode, which means the density is less than a specified critical density ρ_c, the density function is given by the following equation $u(x,t)_t + v_1 u(x,t)_x = 0$, where v_1 is always a positive number and denotes that the density propagates forward. However, if the traffic density achieves ρ_c, the system transitions into a high density congestion mode. In this mode, the governing equation is described by $u(x,t)_t + v_2 u(x,t)_x = 0$, where v_2 is always a negative number and denotes that traffic congestion always move backwards and propagates upstream. As an example, one of the causes of congestion may be a result of merging slow and fast moving traffic resulting in a high density traffic flow. In this model, we can frequently switch between a free flowing mode and a congestion mode, and as a result, a more powerful hybrid framework is needed to formalize these kinds of system behavior.

Additionally, the two models we have outlined present a mixture of space and time variables and are therefore quite complicated. In the following sections we present a framework that is able to capture such a mixture of discrete and continuous behaviors, where PDEs describe the continuous dynamics.

3.4 Partial Differential Hybrid Automata

3.4.1 Transitions, Trajectories and Executions

Due to the properties of PDEs, the switching structure is more involved than in the case of ODEs [44]. One highway congestion alleviation model in [19] considers N connected segments where each segment is governed by an independent LWR PDE [118] with one side boundary condition. Each segment will affect the boundary condition of the proceeding segment and serves as a input function to the PDE. Additionally, the system is allowed to switch into another mode that incorporates a speed limit. One control approach for transport systems [83] proposes rules for switching based on a criteria of minimizing a cost function. We outline two specific structures that are of interest to our framework:

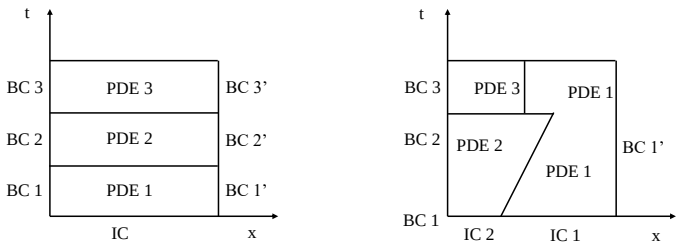


Figure 3.3: Switching cases for two scenarios, full domain switching on left and partial domain switching on right.

- *Switching PDEs in time on the full spatial domain*. This situation is portrayed in Figure 3.3 (left) and is the PDE counterpart of hybrid systems as defined in [183], where a game theory approach is used to determine when mode switching occurs. Boundary condition switching has was investigated in highway traffic applications in [160]. PDE switching in time appears in [83] and in [19] and the latter one contains switching between modes in each highway segments.

- *Switching PDEs on subsets of the time-space domain*. This is illustrated in Figure 3.3 (right). Examples include the LWR PDE, in [19] where each highway segment is coupled with a boundary condition at the starting point of itself. This also appears in [118], where the LWR with triangular flux function can be decomposed into two modes (two one-dimensional wave equations) resulting in a partition of the (x,t) space in regions with forward traveling waves, and regions with backward travelling waves.

In order to describe the partial domain switching case in Figure 3.3 (right), it is useful to partition the whole domain into several parts and assign each one a corresponding mode. One possible solution is to use the union of disjoint domains to represent the region governed by specific modes. For example, the heat equation $u_t - u_{xx} = 0$ with $\alpha = 1$ (mode 1) and without heat source can cover $[0,0.3) \cup (0.7,1]$. The equation $u_t - u_{xx} = f(x)$ with external source (mode 2) occupies $[0.3,0.7]$.

Based on this idea, the main challenge is how to formalize the particular structure of the domain occupying behavior. One approach is to associate the union operator with the modes of the system. Thus the system mode q can be described as a composition of multiple modes $\cup q_i$, where each mode specifies the domain it occupies. Like in q_1, $u_t - u_{xx} = 0$ when $x \in \mathscr{D}_1$ and in q_2, $u_t - u_{xx} = f(x)$ when $x \in \mathscr{D}_2$, where $\mathscr{D}_i$ is the region occupied by q_i. Thus, the space domain X is composed of $\mathscr{D}_1$ and $\mathscr{D}_2$.

This embodies our approach to represent the multiple mode occupation phenomenon and we refer to the multiple mode occupation as the changing behavior of $\cup \mathscr{D}_i$. The remaining challenge is to how to describe the moving behavior of the modes. This can be realized by changing $\mathscr{D}_i$, together with its boundary condition. As the system evolves, points satisfying a given condition will be moved from one mode to another and cause

$\mathscr{D}_i$ to grow or shrink. We call this point's moving behavior a *trajectory*. The evolution of $\mathscr{D}_i$ is not easily depicted. It can be a function of time, or a function of several other complicated variables. Additionally, it might be impossible to predict switching behavior due to the complexity exhibited by the PDE's solution.

Based on the ideas discussed above, the concept of *space control mode* is given below.

Definition 1 (Space Control Mode)**.** *Let X be a space domain and $X \subseteq \mathbb{R}^n$. Let Q be a finite set of m control modes where each mode is denoted as q_i. A space control mode consists of a union of k $(k \leq m)$ control modes where each mode covers part of the space domain X. The state control mode is denoted by $\tilde{q}$ where $\tilde{q} = \cup_{q_i \in Q}\{q_i\} \subseteq Q$ and the region mode q_i covers is denoted by $Dom(q_i)$ and $Dom(\cdot)$ is a set map $Dom(\cdot) : Q \to 2^X$. Additionally, $\cup_{i=1}^{k} \mathscr{D}_i = X$, where $\mathscr{D}_i = Dom(q_i)$.*

Now, according to our definition, $\tilde{q}$ is a subset of Q and not just an element in Q. We want to emphasize that the system could occupy several control modes simultaneously rather than stay at a single mode as defined in classical hybrid automata. In order to accommodate $\tilde{q}$ and $Dom(q_i)$ and for the purposes of formalizing domain division, we rely on *partition* defined in Definition A.1. One can see that $\cup\{\mathscr{D}_i\}$ is a *partition* of domain X since each $\mathscr{D}_i$ is nonempty, $\bigcup\{\mathscr{D}_i\} = X$ and $\mathscr{D}_i \cap \mathscr{D}_j = \emptyset$ for $i \neq j$. For convenience, we denote $p = \cup\{\mathscr{D}_i\}$ a specific *partition* of X consisting of $\mathscr{D}_i$s and denote $P = \cup\{p\}$ a set containing all possible *partition*s of domain X.

It is worth noting that the new PDE dynamic hybrid automaton needs a function to describe its state value. The additional spatial variable requires us to specify the values at different space locations while, in classical hybrid automata, the system only has finite values since the system only considers finite continuous state variables. The values of these state variables and discrete modes can be determined as long as the time is fixed and the system is deterministic. In our model, we denote $u(x)$ the value at current location x of the PDE dynamic automaton and $u(x)$ is a real function defined on X. Also $u(x) \in \mathbb{R}^X$ where $\mathbb{R}^X$ is the real function space defined on domain X and is an extension to $\mathbb{R}^n$ which is used in classical hybrid automatoa. We denote this by $U = \mathbb{R}^X$. In our framework, we only consider one PDE dynamic equation but we allow for unlimited spatial dimensions. In the real world, 3 dimensions are sufficient for simulation.

After we obtain $\tilde{q}$, p and $u(x)$, the *state* can be defined now.

Definition 2 (State)**.** *A state s is a collection of variables that describes the current condition of a PDE dynamic system. It consists of a* space control mode *$\tilde{q}$, a* partition *p of a domain X and a real function $u(x)$ defined on X.*

$$s = \tilde{q} \times p \times u(x) \in 2^Q \times P \times U \tag{3.6}$$

In Figure 3.3 (left), each mode occupies the whole space domain X and the system stays in only one mode. We have $q = \{q_i\}, \mathscr{D}_i = X$, $i = 1,2,3$. In Figure 3.3 (right), PDE 1 (mode 1) and PDE 2 (modes 2) occupy the whole domain X, $q = \{q_1 \cup q_2\}$ and $\mathscr{D}_1 \cup \mathscr{D}_2 = X$ at $t = 0$. It is worth stressing the differences between representing a state in PDE dynamic hybrid automata and classical hybrid automata. For discrete modes, the combination of $\tilde{q}$ and p replaces the single q and describes the space occupation behavior. For continuous variables, a function $u(x)$ is utilized to specify the value at location x instead of a vector.

Bearing the above in mind, the next step is to formalize the flow function.

Definition 3 (Flow)**.** *A flow for a PDE dynamic hybrid automaton is of the form* $2^{\mathcal{Q}} \times P \times U \to U$ *which maps from one state to another as time evolves.*

Similar to classical hybrid automata, PDE equations simulate the dynamics of the system. The flow consists of a piecewise PDE equation and the corresponding domain partition. A simple flow function for the heater example can be formally represented as

$$\begin{cases} u_t - u_{xx} = 0,\ 0 \le x < c, \\ u_t - u_{xx} = f(x),\ c \le x \le L, \\ u(0,t) = 0, \\ u(L,t) = 0, \end{cases}$$

where c is a constant.

Definition 4 (Transition)**.** *A transition in a PDE dynamic hybrid automaton is of the form* $2^{\mathcal{Q}} \times P \times U \times E \to 2^{\mathcal{Q}} \times P$ *which maps current mode and partition to another.*

In order to illustrate the idea of transition, we make a simple modification to the previous example:

$$\begin{cases} u_t - u_{xx} = 0,\ 0 \le x < ct, \\ u_t - u_{xx} = f(x),\ ct \le x \le L, \\ u(0,t) = 0, \\ u(L,t) = 0. \end{cases}$$

Here, we note that the boundary of the mode OFF $u_t - u_{xx} = 0$ moves right at a rate ct that is proportional to time and the left boundary of the mode ON moves left at the same speed. The space control modes $\tilde{q}$ remain the same and p keeps changing as time advances. Additionally, 0 and L denote the left and right boundary of the heater respectively.

Other types of transitions that may involve event or guard conditions are also included:

$$\begin{cases} u_t - u_{xx} = 0,\ 0 \le x < L/2, \\ u_t - u_{xx} = f(x),\ L/2 \le x \le L, \\ u(0,t) = 0, \\ u(L,t) = 0. \end{cases} \xrightarrow{e} \begin{cases} u_t - u_{xx} = f(x),\ 0 \le x < L, \\ u(0,t) = 0, \\ u(L,t) = 0. \end{cases}$$

As long as the event e is enabled, the heater on the left half side will switch from OFF to ON. The event condition can be replaced by a guard condition which guarantees the transition as well.

Definition 5 (Reset)**.** *A reset function for a PDE dynamic hybrid automaton is of the form* $2^Q \times P \times U \times E \to U$ *which specifies how the value of* $u(x)$ *changes when a transition takes place.*

3.4.2 Partial Differential Hybrid Automaton (PDHA)

With the above discussion in mind, we can now provide a formal definition of a partial differential hybrid automaton.

Definition 6 (Partial Differential Hybrid Automaton (PDHA))**.** *A* partial differential hybrid automaton *is a tuple* $\langle Q, E, P, X, U, Init, Inv, f, \phi, G, R \rangle$ *where:*

- Q *is a set of finite* q_i*s that represents control modes.*
- E *is a finite set of events.*
- X *is a space domain and* $X \subseteq \mathbb{R}^n$.
- P *is a set of all partitions defined on domain* X.
- U *is a function space defined on domain* X *and* $U = \mathbb{R}^X$.
- $Init$ *is a set of initial states and* $Init \subseteq 2^Q \times P \times U$.
- Inv *is a set of invariants defined for each mode* q_i.
- $f : 2^Q \times P \times U \to U$ *is a flow function and* f *has the form :*

$$\begin{cases} u_t = f_1(\Delta u, \nabla u, x, t),\ \ x \in \mathscr{D}_1, \\ u_t = f_2(\Delta u, \nabla u, x, t),\ \ x \in \mathscr{D}_2, \\ \ldots \\ u_t = f_k(\Delta u, \nabla u, x, t),\ \ x \in \mathscr{D}_k. \end{cases}$$

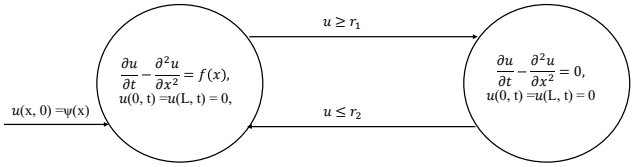


Figure 3.4: Heater PDHA corresponding to the system in Figure 3.1.

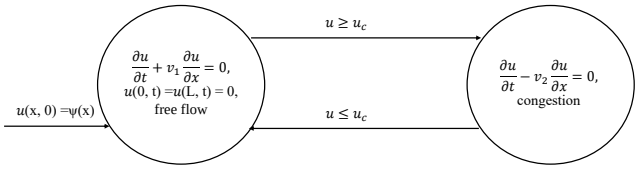


Figure 3.5: Traffic PDHA corresponding to the system in Figure 3.2.

where $f_i, i \in \{1,2,...,k\}$*, is the continuous segment in* f*,* Δ *is the Laplacian,* ∇ *is the gradient, and* $\mathscr{D}_i, i \in \{1,2,...,k\}$ *is the domain covered by* q_i*.*

- $\phi : 2^Q \times P \times U \times E \to 2^Q \times P$ *is a transition map.*
- G *is a set defining guard condition,* $G \subseteq 2^Q \times P \times 2^Q \times P \times U$*.*
- $R : 2^Q \times P \times U \times E \to U$ *is a reset function.*

In our framework, each mode contains only one PDE instead of a system of equations as in classical hybrid automata. Boundary conditions must not be violated and can be treated as a kind of *invariant*. They are imposed on the boundary of X and act like constraints within the mode. Additionally they are forced to be satisfied even during mode discrete transitions.

Revisiting the examples outlined at the very beginning, we construct two models to illustrate the idea of PDHA in Figures 3.1 and 3.2. In each model, the system is composed of two modes and each mode is governed by a PDE, the given boundary conditions, and the initial condition. In the heater model, we extract the term Δu from $f_i(\Delta u, \nabla u, x, t)$ and use $f_i(x)$ to represent the input function.

The continuous trajectory is realized by moving a point satisfying a guard condition G to another mode. In the heater model, a point in the mode ON where the temperature exceeds the threshold will be transferred to the mode OFF, meaning the heater located at this point is turned off while the other areas remain unchanged. At the same time, as the temperature at some points drops below some given bound, the heater at that location is turned on and the point at mode OFF will be moved to mode ON.

The same rule applies to the traffic model as well. The points in the free flow mode satisfying the guard condition G_1 will be moved to a congestion mode and points in the congestion mode satisfying G_2 will be

moved to the free flow mode.

3.5 Discrete Space Partial Differential Hybrid Automata

In this section we describe some core steps in deriving DSPDHA. First, we define a discretization scheme and use it to discretize a PDHA to construct its "discrete" model for analysis. Second, using this discretization scheme, we define a discrete PDHA that is related to the original PDHA via a "discretization relation" corresponding to the scheme $\mathscr{R}$.

3.5.1 Discretization Scheme and Discretization Relation

A discrete treatment of PDEs requires the use of new tools to realize the functions we need. One of the most widely used set of methods are called finite difference methods (FDMs). Generally speaking, FDMs are a family of numerical approaches for solving differential equations by replacing them with an approximation using difference equations. Combining FDMs and PDHA, produces the following definition:

Definition 7 (Discretization Scheme)**.** *A discretization scheme is a numerical scheme to discretize a continuous partial differential equation into a set of algebraic equations (full-discretization scheme) or ordinary differential equations (semi-discretization scheme).*

We present several schemes below to demonstrate how this technique approximates spatial derivatives.

Example 3.5.1 (Finite Difference Method Discretization Scheme)**.** *A common example is shown for the heat equation using the central difference scheme:*

$$\begin{aligned} u(i\times h,t)_{xx} &= \frac{u((i-1)\times h,t)-2u(i\times h,t)+u((i+1)\times h,t)}{h^2} \\ &+O(h^2), \end{aligned} \tag{3.7}$$

where the scheme is based on uniform mesh, h is the grid size, $h = x_{i+1} - x_i$, $\{x_1, x_2, x_3, ..., x_n\}$ is a list of locations where the mesh points sit, and these points are the same ones defined in p_d. Additionally, $u(i\times h,t)$ denotes the value at the i^{th} position given time t and the respective derivative is approximated using itself and the two adjacent values. The proof is based on using Taylor's expansion with error $O(h^2)$.

After applying (3.7) and dropping the redundant part $O(h^2)$, the heat equation becomes

$$\dot{u_i}(t) \approx \frac{1}{h^2}(u_{i-1}(t)-2u_i(t)+u_{i+1}(t))+f(i\times h,t). \tag{3.8}$$

We replace $u(i\times h,t)$ by $u_i(t)$ since the derivative with respect to space has been removed. Thus, $\dot{u_i}(t)$ is used to represent the time derivative at grid point $i\times h$, and $f(i\times h,t)$ represents input function valued at the

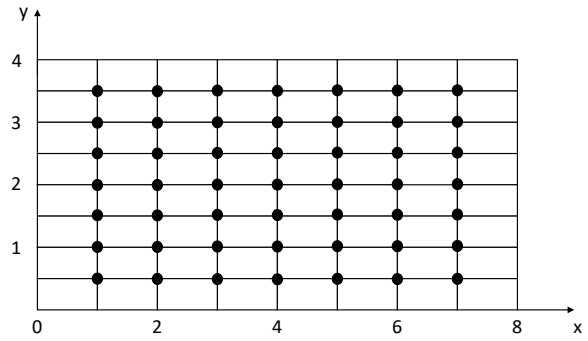


Figure 3.6: Regular mesh on domain $X(0 \leq x \leq 8 \wedge 0 \leq y \leq 4)$, black dots represent the points chosen in X_d.

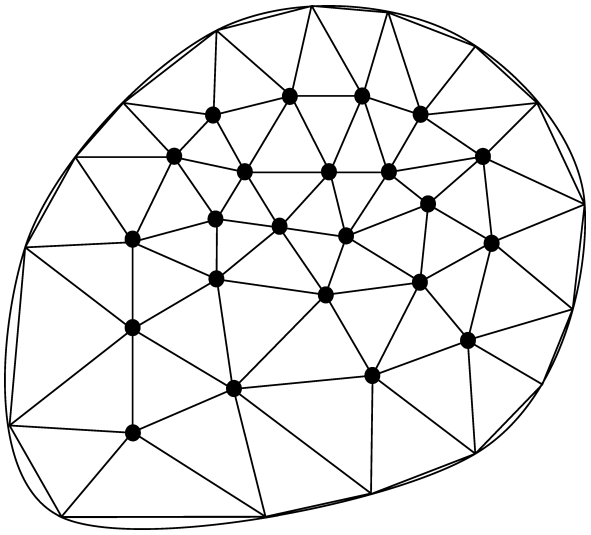


Figure 3.7: Irregular mesh on domain X, black dots represent the points chosen in X_d.

same location.

Once the scheme has beeb chosen, the relation between our models can be described by the following definition.

Definition 8 (Discretization Relation)**.** *If model A is obtained from model B via discretization scheme $\mathscr{R}$, we say that the model A relates to model B via discretization scheme $\mathscr{R}$, which is denoted by: $A \preceq^{\mathscr{R}} B$.*

3.5.2 Discrete Space Partial Differential Hybrid Automaton

Using a discretization scheme, we obtain discrete models of the state, transition, flow and reset function of the original PHDA. These discrete models are defined below.

Definition 9 (Discrete Domain)**.** *Let X be a domain and $X \subseteq \mathbb{R}^n$, a discrete domain X_d of X is a set of distinct points obtained from X by a discretization scheme $\mathscr{R}$: $X_d = \bigcup_{i=1}^{m} \{x_i\}$, $x_i \in X$ and m is the number of points. We say $X_d \preceq^{\mathscr{R}} X$.*

Example 3.5.2 (Discrete Domain of Heater Model)**.** *Consider the heat equation defined on domain* $X = [0,10]$. *One possible corresponding discrete domain* X_d *is* $\{0,1,2,3,4,5,6,7,8,9,10\}$.

Definition 10 (Discrete Partition)**.** *Let* Q *be a set of modes,* P *is the set of all partitions on* X *and* $X_d \preceq^{\mathscr{R}} X$. *A discrete partition* $p_d \in Q^m$ *is a set of values and each value is associated with a point* $x_i \in X_d$. *The set of all possible* p_d*s is denoted by* $P_d = Q^m$ *and* $P_d \preceq^{\mathscr{R}} P$.

In contrast to the continuous domain hybrid automaton which uses $\tilde{q} \times p$ to describe the current mode that the system occupies, a discrete partition combines the two notations and simplifies the expression.

Example 3.5.3 (Discrete Partition of Heater Model)**.** *Consider the model with* $q_1 = OFF$, $q_2 = ON$ *and* $X_d = \{0,1,2,3,4,5\}$. *Then* $\tilde{q} = \{q_1, q_2\}$, $\mathscr{D}_1 = [0,3]$ *and* $\mathscr{D}_2 = (3,5]$ *can be expressed as* $\{q_1, q_1, q_1, q_1, q_2, q_2\}$.

Definition 11 (Discrete State)**.** *A discrete state* s_d *is a ordered pair* (p_d, u_d) *where* $p_d \in P_d \preceq^{\mathscr{R}} P$ *is a discrete partition on* $X_d \preceq^{\mathscr{R}} X$ *and* $u_d \in \mathbb{R}^m$ *is a list of real values. We denote the set of all possible* s_d*s by* S_d *and say* $S_d \preceq^{\mathscr{R}} S$ *where* S *is the set of states defined in PDHA.*

A discrete state gives the whole picture of the system by combining a discrete partition and state values.

Example 3.5.4 (Discrete State of Heater Model)**.** *One discrete state can be in the form* $\{\{q_1, q_1, q_1, q_1, q_2, q_2\}$, $\{0.6, 0.6, 0.5, 0.6, 0.7, 0.8\}\}$, *where* q_1, q_2 *are the modes and the succeeding numbers are the temperatures at each node.*

Everything listed above leads us to *discrete flow*.

Definition 12 (Discrete Flow)**.** *Let* f *be a flow function ,* f_d *is the set of discrete flow functions such that the spatial derivative of* f *are approximated using* $\mathscr{R}$. *We say* $f_d \preceq^{\mathscr{R}} f$.

Example 3.5.5 (Discrete Flow of Heater Model)**.** *Consider the heater example again,*

$$\begin{cases} u_t - u_{xx} = 0,\ 0 \le x < c, \\ u_t - u_{xx} = f(x),\ c \le x \le L, \\ u(0,t) = 0, \\ u(L,t) = 0. \end{cases}$$

The spatial derivative u_{xx} *is approximated by a FDM centered difference scheme based on a uniform mesh. We obtain a system of ODEs:*

$$\begin{cases} \dot{u_1} = \frac{-2u_1+u_2}{\Delta x^2},\ x = x_1, \\ \dot{u_2} = \frac{u_1-2u_2+u_3}{\Delta x^2},\ x = x_2, \\ \ldots \\ \dot{u_{c-1}} = \frac{u_{c-2}-2u_{c-1}+u_{c+1}}{\Delta x^2},\ x = x_{c-1}, \\ \dot{u_{c+1}} = \frac{u_{c-1}-2u_{c+1}+u_{c+2}}{\Delta x^2} + f(x_{c+1}),\ x = x_{c+1}, \\ \ldots \\ \dot{u_m} = \frac{u_{m-1}-2u_m}{\Delta x^2} + f(x_m),\ x = x_m. \end{cases}$$

where x_{c-1} and x_{c+1} are the nearest two mesh points to c. The boundary conditions are merged into the first and last equations.

The flow function can assume a complicated form, making the semi-discretization of the PDE difficult. This represents an interesting arena for further investigation that we will consider in the future.

Definition 13 (Discrete Transition)**.** *A discrete transition ϕ_d of a DSPDHA is of the form $P_d \times U_d \times E \to P_d$ which maps current discrete partition to another. We say $\phi_d \preceq^{\mathscr{R}} \phi$ where ϕ is the transition defined in the original PDHA and $\mathscr{R}$ is the discretization scheme.*

Example 3.5.6 (Discrete Transition of heater model)**.** *In order to illustrate the idea of a discrete transition, we make a simple modification of the previous example:*

$$\begin{cases} \dot{u_1} = 0,\ \ x = x_1, \\ \dot{u_2} = 0,\ \ x = x_2, \\ \ldots \\ \dot{u_m} = f_m(u_1, u_2, \ldots, u_m, t),\ \ x = x_m. \end{cases}$$

$$\xrightarrow{e'} \begin{cases} \dot{u_1} = f_1(u_1, u_2, \ldots, u_m, t),\ \ x = x_1, \\ \dot{u_2} = f_2(u_1, u_2, \ldots, u_m, t),\ \ x = x_2, \\ \ldots \\ \dot{u_m} = f_m(u_1, u_2, \ldots, u_m, t),\ \ x = x_m. \end{cases}$$

A transition to the mode ON at location x_1 and x_2 occurs because event e takes place. Thus, we can write

the transition function

$$\phi_d(p_{d,1};u_d;e) = \begin{cases} p_{d,2}, & e = e', \\ p_{d,1}, & otherwise. \end{cases} \tag{3.9}$$

where $p_{d,1} = \{q_1, q_1, q_2, ..., q_2\}$, $p_{d,2} = \{q_2, q_2, q_2, ..., q_2\}$, q_1 *represents mode OFF and* q_2 *represents mode ON.*

Definition 14 (Discrete Reset)**.** *A discrete reset function* R_d *of a DSPDHA is of the form* $P_d \times U_d \times E \to U_d$ *which specifies how a value of* u_d *changes to a new value when a transition takes place. We say* $R_d \preceq^{\mathscr{R}} R$ *where R on the right is the reset function in original PDHA.*

Definition 15 (Discrete Space Partial Differential Hybrid Automaton (DSPDHA))**.** *A* discrete space partial differential hybrid automaton *is a tuple* $\langle Q, E, P_d, X_d, U, Init,$
$Inv, f_d, \phi_d, G, R_d \rangle$ *where:*

- Q *is a set of finite modes;*
- E *is a set of finite events;*
- X_d *is a set of m points;*
- P_d *is a set of values defined on domain* X_d, $P_d \in Q^m$*;*
- U_d *is a set of discrete values defined on* X_d, $U_d = \mathbb{R}^m$*;*
- $Init$ *is a set of initial states,* $Init \subseteq P_d \times U_d$*;*
- Inv *is a set of invariants defined for each mode* $q_i \in Q$*;*
- $f_d : P_d \times U_d \to U_d$ *is a set of discrete flow functions and* f_d *have the form :*

$$\begin{cases} \dot{u_1} = f_1(u_1, u_2, ..., u_m, t), & x = x_1, \\ \dot{u_2} = f_2(u_1, u_2, ..., u_m, t), & x = x_2, \\ ... \\ \dot{u_m} = f_m(u_1, u_2, ..., u_m, t), & x = x_m. \end{cases} \tag{3.10}$$

  where $\dot{u}_i$ *denotes time derivative of* u_i*;*

- $\phi_d : P_d \times U_d \times E \to P_d$ *is a transition function;*
- G *is a set defining guard condition,* $G \subseteq P_d \times P_d \times U_d$*;*

- $R_d : P_d \times U_d \times E \to U_d$ *is a reset function.*

A key observation is $DPH \preceq^{\mathscr{R}} PH$ if the discrete space partial differential hybrid automaton *DPH* is obtained from a partial differential hybrid automaton *PH* through $\mathscr{R}$.

The DSPHDA provides us with a set of properties that can easily be formalized and stated. Additionally, similar concepts that appear in HA are outlined below.

Definition 16 (Hybrid Time Trajectory)**.** *A hybrid time trajectory* $\tau = \{I_i\}_{i=0}^{N}$ *is a finite or infinite sequence of intervals of the real line, such that*

- $I_i = [\tau_i, \tau_i^{'}]$ *for* $i < N$*;*
- *if* $N < \infty$*, then either* $I_N = [\tau_N, \tau_N^{'}]$*, or* $I_N = [\tau_N, \tau_N^{'})$*;*
- *for all i,* $\tau_i \leq \tau_i^{'} = \tau_{i+1}$.

where $\tau_i^{'}$ are the time when *discrete transition* takes place.

Now that we have defined the syntax of DSPDHA, we define the semantics in terms of *executions*.

Definition 17 (Execution)**.** *An* execution *of a DSPDHA is a collection* $\chi = (\tau, p_d, u_d)$ *satisfying*

- *Initial Condition:* $(p_d(\tau_0), u_d(\tau_0)) \in Init$*;*
- *Continuous Evolution: for all i with* $\tau_i < \tau_i^{'}$*,* p_d *is a constant,* u_d *is a solution to the ODEs (3.10) over* $[\tau_i, \tau_i^{'}]$ *and for all* $t \in [\tau_i, \tau_i^{'})$*,* $u_d(t) \in I$*;*
- *Discrete Evolution: for all i,* $(p_d(\tau_{i+1}), u_d(\tau_{i+1})) \in R(p_d(\tau_i^{'}), u_d(\tau_i^{'}))$.

An execution $\chi = (\tau, p_d, u_d)$ is called *finite* if χ is a finite sequence ending with a closed interval, *infinite* if τ is either an infinite sequence, or if $\sum_{i=0}^{\infty}(\tau_i^{'} - \tau_i) = \infty$, and Zeno if it is *infinite* but $\sum_{i=0}^{\infty}(\tau_i^{'} - \tau_i) < \infty$.

Definition 18 (Zeno Execution)**.** *An execution* χ *is Zeno if*

$$\lim_{i\to\infty} \tau_i - \tau_0 = \sum_{i=0}^{\infty}(\tau_i^{'} - \tau_i) = \tau_\infty < \infty \tag{3.11}$$

Here τ_∞ *is called the Zeno time.*

We use $\mathscr{E}_{DPH}(p_{d,0}, u_{d,0})$ to denote the set of all executions of DSPDHA with initial condition $(p_{d,0}, u_{d,0}) \in Init$, $\mathscr{E}^{*}_{DPH}(p_{d,0}, u_{d,0})$ to denote the set of all finite executions, $\mathscr{E}^{\infty}_{DPH}(p_{d,0}, u_{d,0})$ to denote the set of all infinite executions, and $\mathscr{E}_{DPH}$ to denote the union of $\mathscr{E}_{DPH}(p_{d,0}, u_{d,0})$ over all $(p_{d,0}, u_{d,0}) \in Init$.

Definition 19 (Zeno Discrete Space Partial Differential Hybrid Automaton)**.** *A discrete space partial differential hybrid automaton DPH is Zeno if there exists* $(p_{d,0}, u_{d,0}) \in Init$ *such that all executions in* $\mathscr{E}^{\infty}_{DPH}(p_{d,0}, u_{d,0})$ *are Zeno.*

$$\lim_{i\to\infty} \tau_i - \tau_0 = \sum_{i=0}^{\infty} (\tau_i^{'} - \tau_i) = \tau_\infty < \infty \tag{3.12}$$

Here τ_∞ *is called the Zeno time.*

The set of states reachable by a DSPDHA, $Reach_{DPH}$, is defined as

$$\begin{aligned} Reach_{DPH} = \{(\hat{p_d}, \hat{u_d}) \in P_d \times U_d : \exists (\tau, p_d, u_d) \in \mathscr{E}^{*}_{DPH}, \\ (p_d(N), u_d(\tau_N^{'})) = (\hat{p_d}, \hat{u_d})\}. \end{aligned} \tag{3.13}$$

where $\tau = \{[\tau_i, \tau_i^{'}]\}_{i=0}^{N}$.

Definition 20 (Nonblocking)**.** *A DSPDHA is called nonblocking if* $\mathscr{E}^{\infty}_{DPH}(p_{d,0}, u_{d,0})$ *is nonempty for all* $(p_{d,0}, u_{d,0}) \in Init$.

Definition 21 (Deterministic)**.** *A DSPDHA is called deterministic if* $\mathscr{E}^{*}_{DPH}(p_{d,0}, u_{d,0})$ *contains at most one element for all* $(p_{d,0}, u_{d,0}) \in Init$.

3.5.3 Relation to Classical Hybrid Automaton

In this section we explore the relations between classical hybrid automata, DSPDHA and PDHA. The central basis of introducing PDHA is extending HA with ODE dynamics to use PDE dynamic. To achieve this goal, some adjustments need to be made to accommodate the new framework within the classical hybrid automata scheme. It is our hope, that after these changes, the properties that hold for HA will hold for PDHA as well. This is the basis for the motivation to add an intermediate level (DSPDHA) lying between PDHA and HA which captures the idea of PDHA and also stays close to classical HA. Using this framework, we make comparisons and illustrate useful results in the form of theorems.

In PDHA, we use space control modes and partitions $\tilde{q} \times p$ to replace a discrete mode q, a state function $\varphi(x)$ to replace state variables X. The weakness of this scheme is determining executions and transitions because $\tilde{q} \times p$ evolve continuously. DSPDHA overcomes these shortcomings by introducing a discrete state partition p_d which functions similar to q in HA. Along the same lines, a discrete flow function obtained from PDHA by FDM is essentially the same as a HA flow function. This allow us to assert some PDE system properties where we cannot otherwise reason with certainty due to a lack of information. The results below shows how these connections are built.

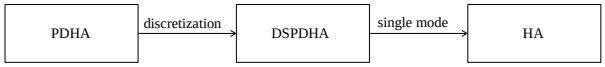


Figure 3.8: PDHA, DSPDHA and HA relations.

Lemma 3.5.1. *A DSPHDA is equivalent to a HA if the DSPDHA stays at a single mode at any time.*

Proof. Based on Definition 15, the given DSPDHA is a HA. □

Lemma 3.5.1 leads us to a more impressive conclusion.

Theorem 3.5.2. *Any DSPDHA is a HA if p_d has identical elements.*

Proof. Based on the knowledge that elements in p_d representing the modes where each location resides, identical elements indicate all the locations are assigned to the same modes and the system stays at a single mode. According to Lemma 3.5.1, the given DSPDHA is a HA. □

Next, the relation between flow functions is explored.

Lemma 3.5.3. *If the order of the time derivative of the PDE considered in a PDHA is* 1*, the number of points m, in a discrete domain X_d is equivalent to the size of flow functions in the corresponding DSPDHA .*

Proof. According to Definition 12, the discretization is made on each mesh point in X_d. As a result, we have m equations eventually. □

Lemma 3.5.4. *If the order of the highest time derivative of a PDE is n, then the PDE can be reduced to a system of n PDEs with a 1^{st} order time derivative.*

Proof. We begin the proof by a change of variables. Let us replace the k^{st} order time derivative of u by v_k and add a new equation to the system once the replacement is done. In the end, we obtain a system of equations in the form below.

$$\begin{cases} \dot{u} = v_1, \\ \dot{v}_1 = v_2, \\ \dot{v}_2 = v_3, \\ \ldots, \\ \dot{v}_{n-2} = v_{n-1}, \\ \dot{v}_{n-1} = f(u, u_x, u_{xx}, ..., t). \end{cases} \tag{3.14}$$

where the original equation is $u^{(n)} = f(u, u_x, u_{xx}, ..., t)$ and $u^{(n)}$ is its n^{th} time derivative. □

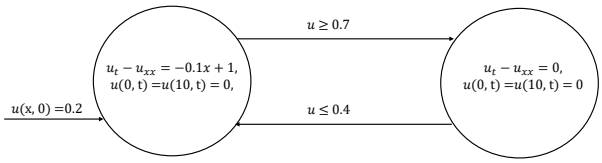


Figure 3.9: PDHA diagram for the heater model analyzed.

Using Lemma 3.5.3 and 3.5.4, a more general argument is given.

Theorem 3.5.5. *If the order of time derivative of the PDE considered in a PDHA is n and the number of points in discrete domain X_d is m, the size of flow functions in the corresponding DSPDHA is $n \cdot m$.*

Proof. Since m determines the number of mesh points and each point is associated with a ODE (1^{st} order PDE after spatial discretization), m equals the dimension of the system of ODEs which is the flow function. □

Besides investigating the relation and distinction between models above, an interesting observation which occurs in classical hybrid automata are discontinuity issues, especially under PDE dynamics. [83] discusses the propagation of discontinuities when the system switches mode. The are numerous reasons why discontinuities occur. For the transport equation with a nonlinear flux function, there does not always exist a smooth solution, and a weak solution has to be constructed to handle a shock or rarefaction wave [182]. Jump discontinuities for hyperbolic system are discussed in [153].

The situation discussed in [83] is likely to happen if the discrete transition take places in our framework. Consider a segment of road governed by a transport equation and the model is $u_t + u_x = 0$. Supposing the switching is triggered and a new model $u_t + 2u_x = 0$ is introduced, the latter model which involves fast wave speed will push the front flow to move fast and the traffic will start to accumulate at the boundary of the road which directly leads to a discontinuity.

3.6 Experiments

The heater and traffic examples are formalized as DSPDHA and analyzed in this section.

Heater Model.

As illustrated above, the first example is the bronze rod with two isolated ends. The heater provides heat to the rod and can be partially turned on and off at any time once the temperature reaches the given thresholds. For the input function, a linear increment function is considered with a maximum value appearing on the left and minimum value appearing value on the right. The diagram of the model is in Figure 3.9.

The idea of *discrete partitions* is illustrated in this the example and 9 points are chosen to describe the temperature along the rod. A point whose value is greater than 0.7 will be moved to mode OFF and be

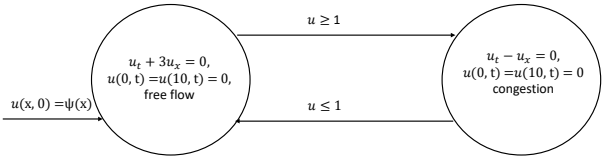


Figure 3.10: PDHA diagram for the traffic model analyzed.

moved back to ON as the value drops below 0.4. The initial temperature is 0.2 everywhere on the rod and the input function for mode ON is $f_1(x) = -0.1x + 1$ and 0 for mode OFF. Since we have the boundary condition $u(0,t) = 0$ and $u(L,t) = 0$, the two values, $u(x_1,t)$ and $u(x_{n-1},t)$, will decrease rapidly due to the heat diffusion effect [58]. Additionally, the input function decreases as x increases, therefore the heating effect near the boundary $x = L$ will be trivial and the heater remains on. We discretize the domain and assign each mesh point a heater. The center difference scheme is used to approximate the space derivative and the forward scheme is used to approximate time derivative.

At $t = 2$, heater at $x = 1$ is OFF and $p_d = \{0,1,1,1,1,1,1,1,1\}$ where 0 represents OFF and 1 represents ON.

Traffic Flow Model.

We consider the model appearing in [44]. The switched PDE problem is a LWR PDE [118] with the triangular flux function. The flux function and two PDE modes are:

$$\phi(u) = \begin{cases} v_1 u, \ x < u_c \\ v_2(\omega - u). \ x \geq u_c \end{cases} \qquad mode = \begin{cases} u_t + v_1 u_x = 0, \ x < u_c \\ u_t - v_2 u_x = 0. \ x \geq u_c \end{cases} \tag{3.15}$$

We chose $v_1 = 3$, $v_2 = 1$, $\omega = 4$ and $u_c = 1$. The forward traffic flow maintains a speed equal to 3 and the backward wave maintains a speed equal to 1. Moreover, u_c equals 1 which is the guard condition for switching. Lastly, ω is the parameter specified in backward wave. The diagram of the model is in Figure 3.10.

Initially, the regions, $\{0,0.1,...,0.8,0.9\} \cup \{1.3,1.4,...,3.3,3.4\} \cup$
$\{3.8,3.9,...,9.9,10.0\}$, are assigned value 0 for free flow mode. The other remaining areas, $\{1.0,1,1,1.2\} \cup$ $\{3.5,3.6,3.7\}$, are assigned value 1 for congestion mode. The Equation 3.15 shows the free flow mode carries the traffic forward and congestion pushes the traffic backward. Once the forward and backward waves meet each other, the forward wave merges into backward wave and starts to move back. The traffic leaves the system and no longer enters again when it reaches the boundary. For simplicity, no incoming traffic is considered.

We can facilitate simulation if we harness the theoretical aspects of linear hyperbolic equations. To simulate the first order wave equation, only one-side scheme is feasible and the low accuracy of the scheme requires a very dense discretization [182] which makes the computation task a nightmare. Thus, the simulation is just done by moving the waves alone the axis.

At $t = 1$, backward wave appearing at boundary $x = 0$ is about to leave. Wave originating from $[3.5, 3.7]$ moves to the place around 2.4. The Forward wave starting at $[3.9, 4.0]$ propagates to region $[6, 9, 7.0]$. The *discrete partition* p_d changes to 0 for points $\{0.3, 0.4, ..., 2.3, 2.4\} \cup \{2.8, 2.9, ..., 9.9, 10\}$ and 1 for points $\{0, 0.1,$

$0.2\} \cup \{2.5, 2.6, 2.7\}$. At $t = 3$, only one single backward wave remains. $p_d = 0$ for $\{0, 0.1, ..., 0.7, 0.8\} \cup [1.1, 1.2, ..., 9.9, 10]$ and $p_d = 1$ for $\{0.9, 1.0\}$.

Simulations show that the traffic merges around $t = 0.5$ and eventually all flows leave the system. Around the merging location, the area that used to be in the free mode switches to the congestion mode.

3.7 Conclusion

In this chapter, we propose a theoretical model called PDHA for modelling PDE dynamic systems. PDHA involves novel features such as the notion of domain X, which is used to define the spatial location, and domain partition P, which is used to describe the mode occupation situation. After building the theory for PDHA, we present another model called discrete space PDHA that we constructed for the purpose of analysis. In discrete space PDHA, the discretization relation and scheme are proposed to formalize the discrete model. Additionally, the partition and space mode are replaced by a discrete partition, making the analysis easier. Moreover, we formally defined concepts such as time trajectory, model execution, Zeno behaviour, reachability analysis, and determinism. The chapter also displays explorations towards the relations of the three models.

CHAPTER 4

Numerical Reachability Analysis for Partial Differential Equations

4.1 Reachability Analysis

Reachability analysis is a solution to the reachability problem in the context of dynamical systems and is the fundamental problem in safety verification of CPS. It computes the set of possible solutions of dynamical systems subject to uncertain initial states and inputs. The problem of systems of linear ODEs has been studied extensively. Set representations include but are not limited to: zonotopes [8], support functions [73], and level sets [129]. Reachability analysis tools like SpaceEx [64] and Flow* [40], which utilize Taylor models, have proven to be powerful. Order reduction abstraction also plays a significant role in evaluating models. The first work combining reachability analysis with order reduction techniques is [81], and a more recent approach is discussed in [188].

4.2 Numerical Reachability Analysis for Hyperbolic Equations

The main contributions of this chapter are novel numerical reachability analysis methods for hyperbolic equations, a particular class of PDEs. We develop numerical reachability analysis for linear/nonlinear hyperbolic equations by combining the flux splitting method, the *Lax-Friedrich* method, and the *Lax-Wendroff* method with optimization and interpolation. These methods provide ways of approximating mesh values and estimating reachable sets on the points. In linear systems, the estimated reachable sets are constructed by computing the trajectories using extreme values from the initial sets. While, in nonlinear systems, an optimization problem is proposed at each time step to calculate the bounds of the values for the next time step. Eventually, the sets are built through interpolation with respect to both time and space, when the values at the current and subsequent time steps are known. In the example section, the estimated reachable sets are compared with real reachable sets. Through dense spatial discretization, the estimated reachable sets can provide a good approximation of the real ones.

4.2.1 Problem Formulation

The following class of systems is considered:

$$\begin{cases} u_t + f(u)_x = 0,\ t \geq 0,\ x \in \mathscr{D} \subseteq \mathbb{R} \\ u(x,0) = \varphi(x) \end{cases}, \tag{4.1}$$

where $u = (u_1, u_2, u_3, ..., u_k)^T$, $u_t = (\frac{\partial u_1}{\partial t}, \frac{\partial u_1}{\partial t}, ..., \frac{\partial u_k}{\partial t})^T$, $f(u) = (f_1(u), f_2(u), ..., f_k(u))^T$ is the flux function, $f(u)_x = (\frac{\partial f_1(u)}{\partial x}, \frac{\partial f_2(u)}{\partial x}, ..., \frac{\partial f_k(u)}{\partial x})^T$, $\varphi(x) = (\varphi_1(x), \varphi_2(x), ..., \varphi_k(x))^T$ is the initial condition, $\mathscr{D} \subseteq \mathbb{R}$ is the domain of interest and $u : \mathscr{D} \times (\mathbb{R}^+ \cup 0) \to \mathbb{R}^k$.

The initial condition is represented by $\varphi(x)$, $\varphi : \mathscr{D} \mapsto \mathbb{R}^k$, and is constrained by a range $[\varphi_l(x), \varphi_u(x)]$, where $\varphi_l(x)$ is the lower bound of $\varphi(x)$ for $x \in \mathscr{D}$ and $\varphi_u(x)$ is the upper bound of $\varphi(x)$ for $x \in \mathscr{D}$. The lower and upper bounds compose the initial set of the problem described in (4.1). The definition, used in this chapter, for initial sets is:

Definition 22 (Initial Set)**.** *Let $\varphi(x)$ be the initial condition of (4.1) and $\varphi_l(x)$ and $\varphi_u(x)$ are the corresponding lower bound and upper bound of $\varphi(x)$ defined on domain $\mathscr{D}$. The initial set of (4.1) is defined as*

$$\Omega \triangleq \left\{ \bigcup_x [\varphi_l(x), \varphi_u(x)] \mid \textit{for } x \in \mathscr{D} \right\}. \tag{4.2}$$

Reachable sets are shown to start from Ω and evolve accordingly.

Definition 23 (Reachable Set)**.** *The reachable set of (4.1) is*

$$R_u(T) \triangleq \left\{ \bigcup_{x,t} u(x,t) \,|\, u(x,t) \textit{ satisfy } (4.1) \textit{ for } x \in \mathscr{D}, t \in [0,T] \right\}. \tag{4.3}$$

Since PDEs are often nonlinear and unsolvable [177], it is almost impossible to compute the exact reachable set $R_u(T)$ for a system of PDEs. Rather than directly computing the reachable set, a more practical and feasible way is to derive an approximation of $R_u(T)$, which is called reachable set estimation. In this chapter, the estimation is based on the finite difference methods.

Finite difference methods (FDMs) are a group of approaches used for approximating real values or variable derivatives of a function on predefined mesh points. The error between the discrete solution and the exact solution is measured through a Taylor expansion. We show how to construct estimated reachable sets $\tilde{R}_u(T)$ after discussing FDMs and pick the most accurate estimation for $R_u(T)$.

The main focus of this work is on safety verification for hyperbolic systems. The safety specification is expressed by a set S, defined on domain $\mathscr{D}$, and describes the safety requirement.

Definition 24. *Consider a hyperbolic PDE system in the form of (4.1), reachable sets $R_u(T)$ and a safety specification S, the hyperbolic PDE system is safe if the following condition is satisfied*

$$R_u(T) \cap \neg S = \emptyset. \tag{4.4}$$

Definition 24 uses reachable sets for safety verification of a system of PDEs although $R_u(T)$ is not obtain-

able due to the unsolvable property mentioned above. One way to circumvent these difficulties is to utilize FDMs since it produces an explicit error estimation. As the number of mesh points increases, the estimated solution will become more accurate. In the next section, the analysis of linear and nonlinear systems is discussed.

4.2.2 Linear System

The flux function $f(u)$ in (4.1) can be decomposed if its Jacobian matrix is not dependent on the unknown variable u. Then, we obtain the system below.

$$\begin{cases} u_t + Au_x = 0,\ t \in \mathbb{R}^+,\ x \in \mathscr{D} \subseteq \mathbb{R} \\ u(x,0) = \varphi(x) \end{cases} \tag{4.5}$$

where A is the Jacobian matrix of $f(u)$ and is a constant matrix.

Transforming the matrix A into a diagonal matrix Λ allows each $u \in u$ to be treated separately. Next, S is multiplied on both sides of $u_t + Au_x = 0$.

$$\begin{aligned} Su_t + SAu_x &= 0, \\ (Su)_t + SAS^{-1}Su_x &= 0, \\ (Su)_t + \Lambda(Su)_x &= 0. \end{aligned} \tag{4.6}$$

Replacing $(Su)_t$ by v_t and $(Su)_x$ by v_x , we have

$$v_t + \Lambda v_x = 0. \tag{4.7}$$

In order to perform reachability analysis, we define monotone to simplify the computing. If our iteration procedure is monotone, we only need to simulate 2 trajectories to construct the reachable sets. An additional CFL condition is required for the monotone property.

4.2.3 Nonlinear System

In the general form shown in system 4.1, if the Jacobian matrix A is a function of u and $f(u)_x = A(u)u_x$, then it is a nonlinear system.

A commonly used approach for simulating a system is the *Lax-Wendroff* method [182] pages 213–216. It is second-order accurate with respect to both space and time. The main issue with numerical reachability analysis using the *Lax-Wendroff* method is that it is not monotone and does not preserve the property. That

fact has been proved by Harten [84]. In order to handle the non-monotone issue, an optimization problem is constructed and solved at each time step. The maximum and minimum values at each mesh point are found and used as the starting values for the next time step.

The goal is to find $u_{min,i}^{n+1}$ and $u_{max,i}^{n+1}$. To find the optimal solution at time t_n, start with the mesh point i at time t_n and set up the optimizations described below generally using the optimization operator $opt \in \{min, max\}$:

$$\begin{aligned} &\text{opt } u_i^{n+1} \\ &\text{s.t. } \textit{Lax-Wendroff scheme}, \\ &u_{i-1}^n \in [u_{min,i-1}^n,\ u_{max,i-1}^n], \\ &u_i^n \in [u_{min,i}^n,\ u_{max,i}^n], \\ &u_{i+1}^n \in [u_{min,i+1}^n,\ u_{max,i+1}^n]. \end{aligned} \tag{4.8}$$

4.3 Numerical Reachability Analysis for Parabolic Equations with Dense Spatial Discretization

In this chapter, we focus on the numerical reachability analysis for parabolic equations. We extend the state of art in numerical reachability analysis of one/two dimension parabolic equations by combining Krylov subspace method/tridiagonal method with numerical PDEs methods including implicit method, Crank-Nicolson (C-N) method, and Alternative Segment Crank-Nicolson (ASC-N) method. These methods are used to compute approximate values on large number of mesh points by solving a high-dimension sparse matrix problem. In particular, for one dimension heat equation model, we propose a serial approach based on Krylov subspace method. For two dimension problem, we propose an novel approaches combining C-N and Krylov subspace method to estimate the reachable sets. Additionally, for one dimension problem, we utilize a more efficient parallel version C-N method, ASC-N, to construct the estimated reachable sets. Finally, we conduct interpolation in both time and space to build the sets once obtaining values at the current and subsequent time step.

4.3.1 One Dimension Problem

The one dimension problem can be described as follows:

$$\begin{cases} u_t = \alpha u_{xx} + \lambda u,\ t \in \mathbb{R}^+,\ x \in \mathscr{D} \subseteq \mathbb{R} \\ u(x,0) = \varphi(x), \\ u(0,t) = 0, \\ u_x(L,t) = 0, \end{cases} \tag{4.9}$$

	m = 1800	m = 3600	m = 7200
direct subspace method	39.75s	193.24s	694.17s
ASC-N(3 threads)	57.28s	210.97s	813.79s
ASC-N(5 threads)	40.45s	99.35s	392.12s

Table 4.1: Comparison of using different numbers of mesh points and threads for model 4.9. 3 threads ASC-N is slower than direct subspace method due to communication overhead. 5 threads ASC-N is significantly faster than serial approach.

where $u_t = \frac{\partial u}{\partial t}$, $u_x = \frac{\partial u}{\partial x}$, $u_{xx} = \frac{\partial^2 u}{\partial x^2}$, α is the heat diffusion parameter, λu is the input function, $\varphi(x)$ is the initial condition, and $\mathscr{D} \subseteq \mathbb{R}$ is the domain of interest.

We can use either Crank-Nicolson(C-N) or Alternating Segment Crank–Nicolson(ASC-N) scheme to solve the target problem. ASC-N is a parallel version of C-N approach and is faster when a large number of cores are used. After spatial discretization, we use Generalized Minimal Residual (GMRES) to solve the resulting algebraic equations. GMRES output an estimated solution to the original problem and thus can be used to bloat the reachable sets. Also, both C-N and ASC-N are monotone.

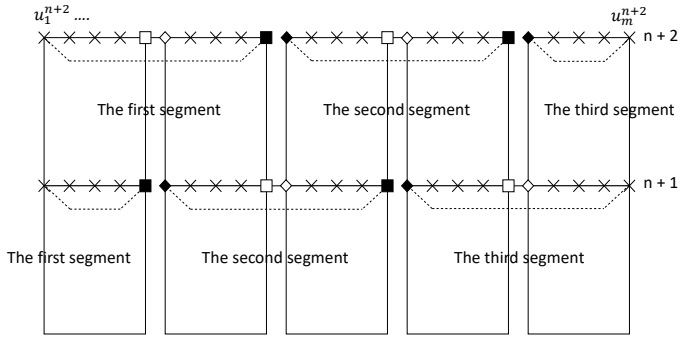


Figure 4.1: The diagram for Alternating Segment Crank-Nicolson(ASC-N). Black squares and diamonds represent the right and left interior boundary points for each segment respectively. White squares and diamonds represent non-boundary points which use to be on boundary.

The numerical reachable analysis of one-dimension system is shown in algorithm 1.

Algorithm 1 Numerical Safety Verification for One Dimension Parabolic Equation

Input: $\mathscr{D}, \Delta t, \Delta x, \lambda_{min}, \lambda_{max}, \alpha, \varphi(x)_l, \varphi(x)_u$
Output: Safe/Uncertain

1: **procedure** CHECK SAFETY
2: Apply C-N or ASC-N to $\varphi(x)_l, \varphi(x)_u$.
3: Solve the equations and bloat the results using GMRES.
4: Construct $\tilde{R}_u(T)$ using interpolation.
5: **if** $\tilde{R}_u(T) \cap \neg S \neq \emptyset$ **then**
6: return Uncertain
7: return Safe

4.3.2 Two Dimension Problem

The two dimension problem is a two dimension heat equation, which can be expressed as follows:

$$\begin{cases} u_t = \alpha(u_{xx} + u_{yy}) + \lambda u,\ t \in \mathbb{R}^+,\ x \in \mathscr{D} \subseteq \mathbb{R} \\ u(x,y,0) = \varphi(x,y), \\ u_y(x,0,t) = 0.5,\ \text{lower boundary} \\ u_y(x,L,t) = 1-u,\ \text{upper boundary} \\ u_x(0,y,t) = 0,\ \text{left boundary} \\ u(L,y,t) = 0.8.\ \text{right boundary} \end{cases} \tag{4.10}$$

We use 2 dimension C-N scheme to discretize the original problem and then apply GMRES to the discrete problem. 2 dimension problem gives us a block tridiagonal matrix which is much larger than one dimension problem matrix.

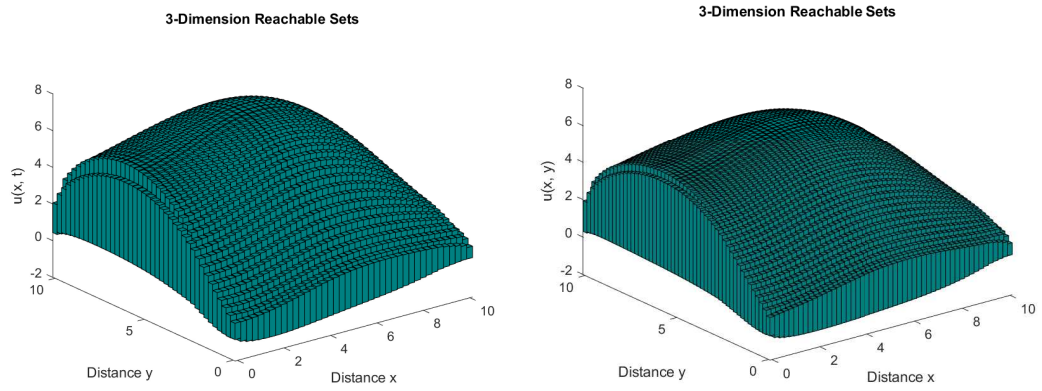


Figure 4.2: 3 dimension estimated reachable sets of two dimension heat equation at $t = 9$ using Crank-Nicolson method scheme with Krylov subspace method. $\Delta t = 0.1$, $\Delta x = 0.4$ and $\Delta y = 0.4$ (left). $\Delta t = 0.1$, $\Delta x = 0.2$ and $\Delta y = 0.2$ (right).

The numerical reachable analysis of two-dimension system is shown in algorithm 2. For the schemes mentioned in above algorithm, see them in Appendix.

Algorithm 2 Numerical Safety Verification for Two Dimension Parabolic Equation

Input: $\mathscr{D}, \Delta t, \Delta x, \Delta y, \lambda_{min}, \lambda_{max}, \alpha, \varphi(x,y)_l, \varphi(x,y)_u$
Output: Safe/Uncertain

1: **procedure** CHECK SAFETY
2: Apply C-N to $\varphi(x)_l, \varphi(x)_u$
3: Solve the equations and bloat the results using GMRES.
4: Construct $\tilde{R}_u(T)$ using interpolation.
5: **if** $\tilde{R}_u(T) \cap \neg S \neq \emptyset$ **then**
6: return Uncertain
7: return Safe

CHAPTER 5

PDE-Driven Neural Networks for Modeling Dynamic Spatial Dependencies

5.1 Modelling Water Temperature using Graph Convolutional Networks

Water temperature prediction in river networks is critical for monitoring aquatic ecosystems by providing important information regarding the habitat for aquatic life and aquatic biogeochemical cycling [31, 122]. Effective temperature predictions are also essential for water management decisions. For example, accurate prediction of water temperature can help water managers optimize the water release from reservoirs to maintain the flow and temperature regimes required for quality downstream habitat.

A river network can be considered as a physical system that has multiple interacting processes. In this problem, multiple river segments are connected to each other, and they can show different thermodynamic patterns driven by differences in catchment characteristics (e.g. slope, soil characteristics) and meteorological drivers (e.g. temperature, precipitation). These segments also frequently interact with each other through the water advected from upstream to downstream segments. Rivers are essentially fluid from a physical perspective, with their spatial and temporal patterns described by PDEs that govern fluid dynamics. For example, traditional fluid models have used the Navier–Stokes equation [18] for simulating fluid dynamics in many applications including aquatic science, hydraulic modeling, weather and climate modeling, ocean currents, and aerodynamics. When modeling temperature dynamics in river networks, these PDEs capture not only the temporal thermodynamics but also the spatial heat diffusion and convection from connected river segments [54]. Furthermore, these PDEs, along with other known physical relationships, have been used to build more complex physics-based models [124, 181] to simulate multiple interacting processes on different variables in a system. However, these equations and physics-based models have limits in their predictions due to approximations and parameterizations used to represent underlying processes.

Recent advances in ML, given their great success in commercial applications, have provided unrealized potential for modeling complex data patterns in scientific problems. The power of these models come from their capacity to extract complex nonlinear patterns from observation data and naturally incorporate spatial and temporal data dependencies. For example, RNN models, which take account of temporal dependencies, have shown extensive applicability in speech recognition and machine translation [77, 128]. CNN-based approaches have shown tremendous success in learning spatial patterns in many computer vision applications [69, 105, 170]. Recently, graph neural network models, e.g., GCNs, have shown a great promise for modeling interactions and similarities amongst multiple objects [115, 121, 215, 216] and also have shown

encouraging results for studying river networks [90, 92, 131].

However, there are several major challenges faced by these existing ML methods when they are directly applied to scientific problems. First, the data available for many scientific problems is far smaller than what is needed for effectively training advanced ML models. In a river network, there are often only a handful of river segments in a network that are monitored due to the high cost associated with data collection. Moreover, despite the promise of existing deep learning techniques, they are not originally designed to exploit the unique characteristics of complex scientific systems. Scientific systems are commonly driven by physical processes and variables that evolve and interact at different spatial and temporal scales. While existing GCN-based methods have shown some success in modeling interactions amongst river segments, they commonly create static graphs based on standard distance metrics (e.g., geographic or stream distances) without fully exploiting the physical characteristics of river segments and also do not capture dynamic interactions over time. In stream networks, the flow of water from one stream segment takes a certain amount of time to reach to another stream segment and this time depends on multiple factors such as the stream morphology, catchment characteristics, and weather patterns. Additionally, the connection strength between two stream segments depends on the velocity and volume of the water flowing through individual streams, which can also vary over time. Hence, the weight of different upstream segments on a downstream segment can vary across time depending on the variation of these physical variables. Additionally, existing GCN-based models extract abstract hidden variables that are propagated over the network, but these hidden variables may fail to represent true underlying physical relationships that govern the interactions, especially given limited training data.

In this chapter, we propose a PDE-guided Dynamic Graph Networks (PDE-DGN) to predict water temperature for all the river segments over a long period. The PDE-DGN captures temporal dependencies with a recurrent layer while also modeling the spatial interactions amongst river segments using dynamic graph structures. Moreover, it incorporates the governing PDE that describes the heat transfer process in the river networks. The PDE represents known physical relationships about dynamic interactions amongst river segments, and thus can be used to guide the design of evolving graph structures. In particular, we use finite difference methods to derive the graph structure from the PDE. The representation of PDEs is also limited in that some physical parameters (e.g., coefficients in PDEs) are unknown and commonly require expensive calibration. In our proposed method, these unknown physical parameters can be estimated together with other neural network parameters efficiently using back propagation.

We implement our proposed method for water temperature prediction using collected data from the Delaware River Basin over 36 years. We demonstrate the superior predictive performance of our proposed method over existing ML methods. Our methods have also been shown to produce better performance when

using limited training data that are sparsely distributed over space (i.e., data are only available from certain stream segments) and time (i.e., data are only available from certain seasons). Moreover, the proposed method has better generalizability when tested in data of different distributions. Our contributions can be summarized as follows:

- We introduce a new dynamic recurrent graph network architecture to model a river network with interacting river segments.
- We leverage the knowledge from the underlying PDE to guide the design of evolving graph structure.
- We evaluate the utility in the context of an ecologically and societally relevant problem of monitoring river networks.

5.2 Modelling Water Temperature using PDE Driven Networks

5.2.1 Problem Definition

Our objective is to model the dynamics of temperature in a set of connected river segments that together form a river network. The connections amongst these river segments can be represented in a graph structure $\mathscr{G} = \{\mathscr{V}, \mathscr{E}, \mathbf{A}\}$, where $\mathscr{V}$ represents the set of N river segments and $\mathscr{E}$ represents the set of connections amongst river segments. Specifically, we create an edge $(i, j) \in \mathscr{E}$ if the segment i is connected to segment j. Because we consider the dynamic interactions amongst river segments, the adjacency matrices $\mathbf{A} = \{\mathbf{A}^1, \mathbf{A}^2, ..., \mathbf{A}^T\}$ represent the adjacency levels amongst all the segments at each time step from $t = 1$ to T. Here a higher value of $\mathbf{A}^t_{ij}$ indicates that the segment i has a stronger influence on the segment j at time t. $\mathbf{A}^t_{ij} = 0$ means there is no edge from the segment i to the segment j at this time. In this chapter, we only consider the evolution of adjacency matrix over time while keeping a static set of river segments (i.e., node set $\mathscr{V}$) and stream connections (i.e., edge set $\mathscr{E}$).

5.2.2 Recurrent Neural Networks and Long-Short Term Memory

The RNN model has been widely used to model the temporal patterns in sequential data. The RNN model defines transition relationships for the extracted hidden representation through a recurrent cell structure. In this work, we adopt the LSTM to build the recurrent layer for capturing long-term dependencies. Each LSTM cell has a cell state $\mathbf{c}^t$, which serves as a memory and allows preserving information from the past. Specifically, the LSTM first generates a candidate cell state $\tilde{\mathbf{c}}^t$ by combining $\mathbf{x}^t$ and the hidden representation at previous time step $\mathbf{h}^{t-1}$, as follows:

$$\tilde{\mathbf{c}}^t = \tanh(\mathbf{W}^h_c \mathbf{h}^{t-1} + \mathbf{W}^x_c \mathbf{x}^t + \mathbf{b}_c). \tag{5.1}$$

where $\mathbf{W}$ and $\mathbf{b}$ are matrices and vectors, respectively, of learnable model parameters. Then the LSTM generates a forget gate f^t, an input gate g^t, and an output gate o^t via sigmoid function $\sigma(\cdot)$, as follows:

$$\begin{aligned}
\mathbf{f}^t &= \sigma(\mathbf{W}_f^h \mathbf{h}^{t-1} + \mathbf{W}_f^x \mathbf{x}^t + \mathbf{b}_f), \\
\mathbf{g}^t &= \sigma(\mathbf{W}_g^h \mathbf{h}^{t-1} + \mathbf{W}_g^x \mathbf{x}^t + \mathbf{b}_g), \\
\mathbf{o}^t &= \sigma(\mathbf{W}_o^h \mathbf{h}^{t-1} + \mathbf{W}_o^x \mathbf{x}^t + \mathbf{b}_o).
\end{aligned} \tag{5.2}$$

The forget gate is used to filter the information inherited from $\mathbf{c}^{t-1}$, and the input gate is used to filter the candidate cell state at t. Then we compute the new cell state as follows:

$$\mathbf{c}^t = \mathbf{f}^t \otimes \mathbf{c}^{t-1} + \mathbf{g}^t \otimes \tilde{\mathbf{c}}^t, \tag{5.3}$$

where $\otimes$ denotes the entry-wise product.

Once obtaining the cell state, we can compute the hidden representation by filtering the cell state using the output gate, as follows:

$$\mathbf{h}^t = \mathbf{o}^t \otimes \tanh(\mathbf{c}^t). \tag{5.4}$$

According to the above equations, we can observe that the computation of $\mathbf{h}^t$ combines the information at current time step ($\mathbf{x}^t$) and previous time step ($\mathbf{h}^{t-1}$ and $\mathbf{c}^{t-1}$), and thus encodes the temporal patterns learned from data.

5.3 Method

In this section, we describe the details of the PDE-DGN method. We start with introducing the dynamic graph model architecture for capturing stream water dynamics and interactions amongst river segments. Then we discuss a new strategy to enforce physical relationships to the dynamic graph structure by leveraging the physical knowledge embedded in known governing PDEs.

5.3.1 Dynamic Recurrent Graph Network

In this section, we describe the details of the PDE-DGN method. We start with introducing the dynamic graph model architecture for capturing stream water dynamics and interactions amongst river segments. Then we discuss a new strategy to enforce physical relationships to the dynamic graph structure by leveraging the physical knowledge embedded in known governing PDEs.

Water temperature in rivers has strong spatial and temporal patterns as a result of heat transfer with climate (e.g., solar radiation) and neighboring river segments [54]. The ML model for river networks also needs to

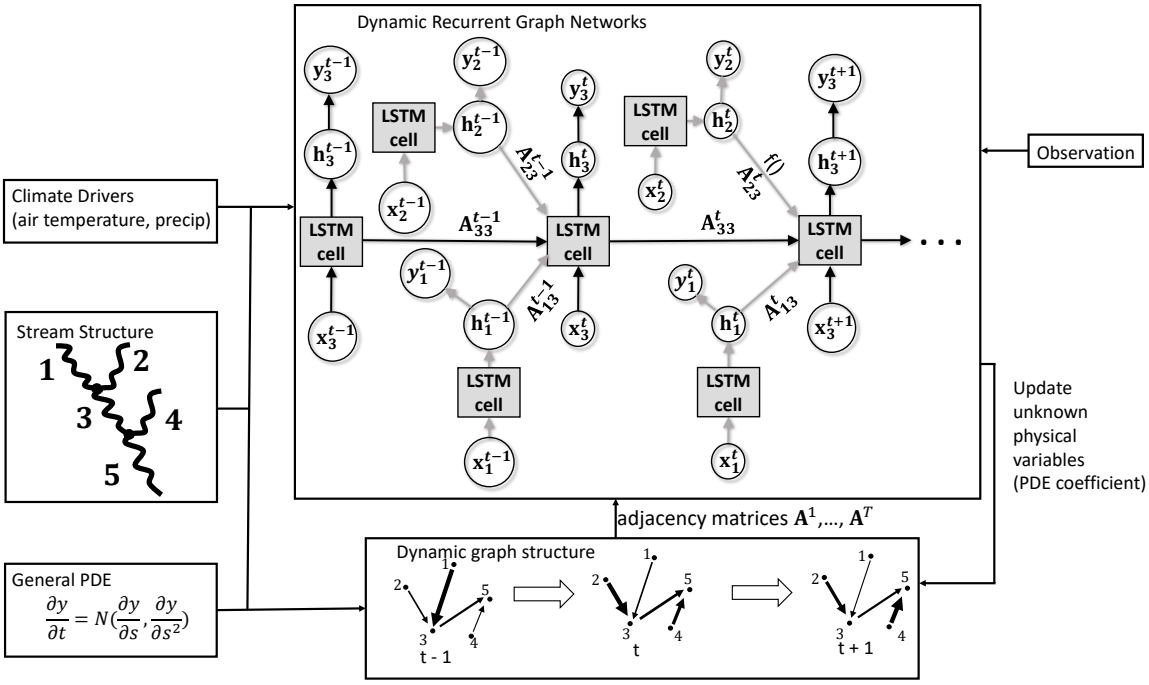


Figure 5.1: The overall flow of the proposed method. The dynamic recurrent graphs take the input of climate drivers and dynamic graph structures derived from the stream network and the underlying PDE. The thickness of edges in dynamic graph structures represents different edge weights. The training of the model updates both network parameters and coefficients in the PDE to refine graph structures.

capture such spatial and temporal dependencies in order to model the temperature dynamics. In particular, we build a dynamic recurrent graph network, which incorporate the information from both previous time steps and neighbors (i.e., upstream segments) when modeling each river segment (Fig. 5.1).

The proposed model aims to embed the input data and the spatio-temporal context of each river segments into a hidden representation $\mathbf{h}^t$ at each time step. Our model structure is inspired by the Recurrent Graph Model [92] but we extend it to take account of changes in the graph structure. We now describe the recurrent process of generating the hidden representation $\mathbf{h}^t$ from $\mathbf{h}^{t-1}$.

First, for each river segment i, the model needs to aggregate the influence from its neighboring segments j such that $(i, j) \in \mathscr{E}$. Specifically, assuming we have gathered the embeddings $\mathbf{h}_j^{t-1}$ from all the upstream segments at previous time step, we first transform these embeddings through a function $f(\cdot)$, which extracts the information that is most relevant to the downstream segment i. For example, the amount of water advected from this segment and its water temperature can directly affect the change of water temperature for its downstream segments. This function can be implemented using fully connected layers. Then we develop a new recurrent cell structure for each segment i by extending the standard LSTM structure (Eq. 5.3) that integrates both the historical information from the same river segment (i.e., the previous state $\mathbf{c}_i^{t-1}$) and the

spatial context from its upstreams (i.e., $f(\mathbf{h}_j^{t-1})$ for $(j,i) \in \mathscr{E}$). This can be expressed as follows:

$$\mathbf{c}_i^t = \mathbf{f}_i^t \otimes (\mathbf{c}_i^{t-1} + \sum_{(i,j)\in\mathscr{E}} \mathbf{A}_{ij}^t f(\mathbf{h}_j^{t-1})) + \mathbf{g}_i^t \otimes \bar{\mathbf{c}}_i^t \tag{5.5}$$

The forget gate not only filters the previous information from the segment i itself but also from its neighbors (i.e., upstream segments). The information from each upstream segment j is weighted by the adjacency level $\mathbf{A}_{ij}^t$ between i and j at time t. The matrix $\mathbf{A}^t$ varies over time due to the change of influence amongst river segments.When a river segment has no upstream segments (i.e., headwater stream segment), the computation of $\mathbf{c}_i^t$ is the same as with the standard LSTM. Also, we use the $f(\mathbf{h}_j^{t-1})$ from the previous time step because of the time delay in transferring the influence from upstream to downstream segments.

After obtaining the cell state, we can compute the hidden representation $\mathbf{h}_i^t$ by following Eq. 5.4. Finally, we generate the predicted output from the hidden representation as follows:

$$\hat{\mathbf{y}}_i^t = \mathbf{W}_y \mathbf{h}_i^t + \mathbf{b}_y, \tag{5.6}$$

where $\mathbf{W}_y$ and $\mathbf{b}_y$ are model parameters.

After applying this recurrent process to all the time steps, we define a loss, $\mathscr{L}_{\text{DGN}}$, using true observations $\mathbf{Y} = \{\mathbf{y}_i^t\}$ that are available at certain time steps and certain segments, as follows:

$$\mathscr{L}_{\text{DGN}} = \frac{1}{|\mathbf{Y}|} \sum_{\{(i,t)|\mathbf{y}_i^t \in \mathbf{Y}\}} (\mathbf{y}_i^t - \hat{\mathbf{y}}_i^t)^2. \tag{5.7}$$

5.3.2 PDE-Driven Dynamic Graph Structure

Water temperature changes in rivers as a result of the heat transfer process. The heat transfer process has been widely studied and also used to build process-based models to simulate water temperature change along stream networks and through time [54]. Here we introduce a new strategy that leverages such underlying physical process to estimate the graph structure used in our proposed method. By enforcing such general physical relationships, the model stands a better chance at learning generalizable patterns even with small amounts of training data.

In particular, we consider the temperature change resulted from the net gain of energy fluxes by following the heat transfer formula described in [54]. This formula is expressed as follows:

$$\frac{\partial y}{\partial t} = -U\frac{\partial y}{\partial s} + D\frac{\partial^2 y}{\partial s^2} + \frac{H_{total}}{\rho \cdot c_p \cdot d}, \tag{5.8}$$

where $y = y(s,t)$ is the water temperature (°C) at time t and location s, U is mean channel velocity (m s^{-1}),

D is a longitudinal dispersion coefficient (m s^{-2}), ρ is the density of water (1000 kg m^{-3}), c_p is the specific heat of water (41.8×10^3 J kg-1 °C^{-1}), and d is the mean channel depth (m). Here $\frac{\partial y}{\partial t}$ denotes the water temperature change over time (s) while $\frac{\partial y}{\partial s}$ denotes the water temperature change over stream distance (m). H_{total} represents the total energy available for transfer to or from the river channel. Eq. 5.8 describes the dynamics of river temperature from a temporal perspective; its rearrangement in the form $\frac{\partial y}{\partial s}$ also permits the calculation of river temperature in a spatial framework [70]. When using Eq. 5.8, we assume that the channel is well mixed and does not contain notable lateral temperature gradients. In order to derive the spatial relationships from Eq. 5.8, we use the finite difference method. Finite difference methods (FDMs) are a group of approaches used for approximating real values or variable derivatives of a function on predefined mesh points by solving algebraic equations containing finite differences and values from nearby points. The error between the discrete solutions and the exact solutions is measured through Taylor expansions.

Our objective is to construct matrix $\mathbf{A}^t$ from the PDE (Eq. 5.8). Real-world river systems are complex. Multiple river segments in a network may interact with each other and these interactions are non-uniform in space and time. Moreover, river segments are usually non-uniformly distributed in space which makes the equal distance FDMs fail. Hence, there is a need to develop a new numerical scheme to handle irregular distributed points. In the following, we will discuss how we use the PDE to contruct the graph structure in two parts: (1) How to numerically approximate the solution of the PDE using irregularly distributed points. (2) How to use use the PDE under special conditions (e.g., intersections and boundaries).

5.3.2.1 PDE over Irregular Points

A feasible way towards the first problem is to use a step variational FDM which is often referred to as generalized finite difference methods (GFDMs). GFDMs are meshless methods and have been used in a wide variety of applications [85]. For the ease of presenting PDEs and GFDMs, we explicitly represent the set of time steps as $\{t_1, t_2, ..., t_T\}$, where the time interval between consecutive time steps is Δt. We also assume the spatial locations of N segments as $\{s_1, s_2, ...s_N\}$. During our presentation, we consider s_{i-1} and s_{i+1} to be the closest upstream and downstream segments to s_i, respectively. Using GFDM, the first order temporal derivative can be approximated as follows:

$$y_t(s_i, t_n) = \frac{\partial y(s_i, t_n)}{\partial t} \approx \frac{y_i(s_i, t_{n+1}) - y_i(s_i, t_n)}{\Delta t}. \tag{5.9}$$

Consider one river segment with the left distance Δs_1 to the left observation point and right distance Δs_2 to the right observation point (see Fig. 5.2). The first order spatial derivative are calculated as follows:

$$y_s(s_{i-1/2},t_n) = \frac{\partial y(s_{i-1/2},t_n)}{\partial s} \approx \frac{y(s_i,t_n) - y(s_{i-1},t_n)}{\Delta s_1}, \tag{5.10}$$

$$y_s(s_{i+1/2},t_n) = \frac{\partial y(s_{i+1/2},t_n)}{\partial s} \approx \frac{y(s_{i+1},t_n) - y(s_i,t_n)}{\Delta s_2}, \tag{5.11}$$

Here, $s_{i+1/2}$ represents the middle points of s_i and s_{i+1}. $s_{i-1/2}$ represents the middle points of s_i and s_{i-1}. We use these two approximated first order spatial derivatives to approximate a second order spatial derivative which can be estimated as follows:

$$y_{ss}(s_i,t_n) = \frac{\partial y_s(s_i,t_n)}{\partial s} \approx \frac{y_s(s_{i+1/2},t_n) - y_s(s_{i-1/2},t_n)}{(\Delta s_1 + \Delta s_2)/2}. \tag{5.12}$$

The truncation error for Eqs. 5.9, 5.10, 5.11 and 5.12 are $O(\Delta t)$, $O(\Delta s_1)$, $O(\Delta s_2)$ and $O(((\Delta s_1 + \Delta s_2)/2)^2)$, respectively.

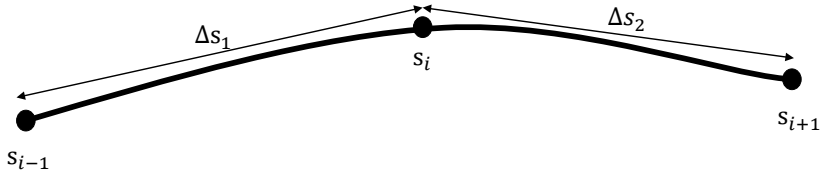


Figure 5.2: The river segment diagram for estimating first-order and second-order spatial derivatives (Eqs. 5.10, 5.11 and 5.12).

Now we can use Eqs. 5.9-5.12 to approximate the derivatives in Eq. 5.8. By substituting the approximated derivatives into Eq. 5.8, we can rewrite the original PDE as follows:

$$\begin{aligned} &\frac{y(s_i,t_{n+1}) - y(s_i,t_n)}{\Delta t} \\ =&-U\frac{y(s_{i+1},t_n) - y(s_i,t_n)}{\Delta s_2} + D\frac{y_s(s_{i+1/2},t_n) - y_s(s_{i-1/2},t_n)}{(\Delta s_1 + \Delta s_2)/2} \\ &+ \frac{H_{total}}{\rho \cdot c_p \cdot d}. \end{aligned} \tag{5.13}$$

Here we use Eq. 5.11 to approximate $y_s(s_i,t_n)$. Then by rearranging the terms in Eq. 5.13, we get the

following representation for $y(s_i, t_{n+1})$, as follows:

$$\begin{aligned} & y(s_i, t_{n+1}) \\ =& (\frac{2\Delta t D}{\Delta s_1(\Delta s_1 + \Delta s_2)}) y(s_{i-1}, t_n) \\ & + (\frac{U\Delta t}{\Delta s_2} - \frac{2\Delta t D}{\Delta s_1(\Delta s_1 + \Delta s_2)} - \frac{2\Delta t D}{\Delta s_2(\Delta s_1 + \Delta s_2)}) y(s_i, t_n) \\ & + (\frac{2\Delta t D}{\Delta s_2(\Delta s_1 + \Delta s_2)} - \frac{U\Delta t}{\Delta s_2}) y(s_{i+1}, t_n) + \frac{H_{total}}{\rho \cdot c_p \cdot d}\Delta t. \end{aligned} \tag{5.14}$$

Because we have multiple rivers in a river graph, we use a vector $Y(t_n) = \{y(s_1, t_n), y(s_2, t_n), ..., y(s_N, t_n)\}$ to represent the temperatures on all the river segments at a specific time step t_n. Note that in Eq. 5.13, $i-1$ and $i+1$ represent two segments that are connected to the segment i, i.e., $(i-1, i) \cup (i, i+1) \subset \mathscr{E}$, and they are the closest upstream and downstream segments to the segment i, respectively. Then we can derive the update formula at each time step by converting Eq. 5.14 into the following form:

$$Y(t_{n+1}) = \mathbf{A}^{t_{n+1}} Y(t_n) + \text{Constant}, \tag{5.15}$$

where each row of $\mathbf{A}^{t_{n+1}}$ can be determined by Eq. 5.14. We ignore the constant term when we build the graph structure. For other variables, the channel velocity U is calculated as the quotient of $\text{streamflow}_i^t / ca_i$, where ca_i represents the cross-sectional area of each stream segment i, and we use the streamflow values that are simulated by a physics-based model PRMS [124]. Because we do not have the measured value of ca_i and D, we estimate these values from observations of water temperature.

5.3.2.2 Dealing with Special Conditions

Now we discuss the estimation of spatial derivatives under two special conditions; intersections and boundaries. First, we consider the intersections in river networks where the target river segment can have multiple upstream segments and one downstream segment (Note that in real-world river networks, it is very rare for naturally flowing river segments to have more than one downstream segment). We show one such example in Fig. 5.3. The calculation of the first-order derivative y_s requires data points from both the target segment and the neighboring segments (Eqs. 5.10 and 5.11). Additional complexity arises if we want to estimate the first-order gradients using multiple upstream segments. Hence, we use the data point from the single downstream segment to estimate the first-order derivative by following Eq. 5.11.

The estimation of second-order derivative y_{ss} (i.e., heat diffusion effect) requires data points from both sides of the target segment. To adapt Eq. 5.12 to handle multiple upstream segments, a straightforward method is to only focus on the closest upstream segment and neglect all the other points, termed as 2-points

approximation. Although this method results in a simplified solution, it may degrade the performance due to the ignorance of the influence from other upstream segments. This issue can be further exacerbated if no observations of water temperature are available for the closest upstream.

Another method is to aggregate the diffusion effect from all the upstream segments. Because more upstream segments lead to more heat exchange, we sum the second-order derivative y_{ss} over all the upstream segments.

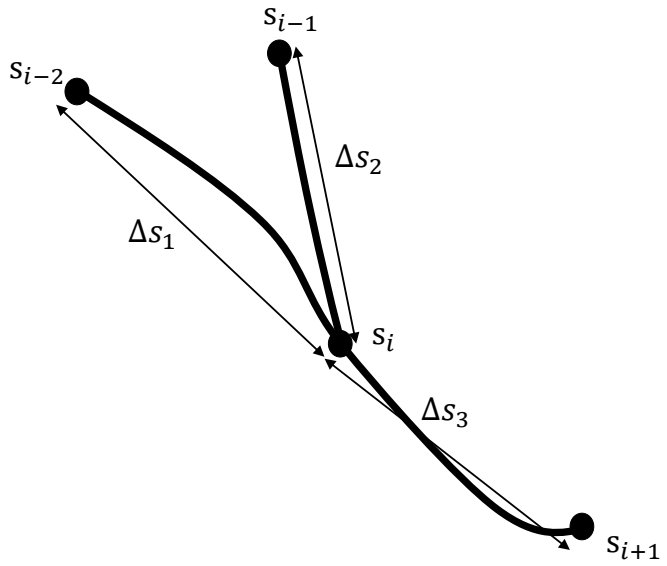


Figure 5.3: River intersection diagram. The center point has two upstream points and one downstream point. In this scenario, we consider $Dy_{ss} = D_1 y_{ss,1} + D_2 y_{ss,2}$.

In particular, in the heat transfer PDE (Eq. 5.8), we convert Dy_{ss} into $\sum_{i=1}^{k} D_i y_{ss,i}$ where k is the total number of upstream segments at the current river segment, and $y_{ss,i}$ represents the second-order derivative estimated using the upstream segment i and the downstream segment following Eq. 5.12. As a result, each row of $\mathbf{A}^t$ contains k values from upstream segments, one value from the downstream segment and one value from itself.

Second, we consider segments that are located at the boundary (i.e., when they have no upstream or downstream segments). It is challenging for estimating spatial derivatives (especially second-order derivatives) for these segments due to the missing neighbors on one side. To this end, we assume the values outside the boundary are always identical to the temperatures on the boundary, i.e., we have the Neumann boundary condition $y_s(s_b,t) = 0$ where s_b denotes a boundary point. Here we show the computation of the second-order derivative for headwater segments (i.e., segments with no upstream segments) as follows:

$$y_{ss}(s_b, t_n) \approx \frac{y_s(s_{b+1/2}, t_n)}{(\Delta s_1 + \Delta s_2)/2}, \tag{5.16}$$

where Δs_1 is a virtual distance measure and is pre-defined in our implementation.

Algorithm 3 Calculation of dynamic graph structures at a specific time step t.

for i = 1 : number of river segments **do**
  U = streamflow$_i^t / ca_i$
  for k = 1 : number of upstreams of i **do**
    $coef_{up,k} = \frac{2\Delta t crossarea_i}{\Delta s_{1,k}(\Delta s_{1,k}+\Delta s_2)}$
    $coef_{i,k} = \frac{U\Delta t}{\Delta s_2} - \frac{2\Delta t D_L}{\Delta s_{1,k}(\Delta s_{1,k}+\Delta s_2)} - \frac{2\Delta t D_L}{\Delta s_2(\Delta s_{1,k}+\Delta s_2)}$
    $coef_{dn,k} = \frac{2\Delta t D_L}{\Delta s_2(\Delta s_{i,k}+\Delta s_2)} - \frac{U\Delta t}{\Delta s_2}$
  $coef_i = \sum_k coef_{i,k}$
  $coef_{dn} = \sum_k coef_{dn,k}$
  for k = 1 : number of upstreams of i **do**
    $A^t_{i,up(k)} = coef_{up,k}$, where $up(k)$ is the index of k^{th} upstream segment
  $A^t_{i,i} = coef_i$
  $A^t_{i,dn} = coef_{dn}$

Algorithm 4 The flow of the proposed PDE-DGN model.

initialize the network model and the physical parameters $\{D_i\}$ and $\{ca_i\}$
for $epoch$ = 1 : number of training iterations **do**
  for t = 1 : number of time steps **do**
    Estimate adjacency matrix A^t using the current values of $\{D_i\}$ and $\{ca_i\}$ following Algorithm 1
    Make predictions using the recurrent graph network following Eqs. 5.5-5.6
    Add the accumulated errors to the loss function (Eq. 5.7)
  update model parameters (i.e., networks weights) and physical parameters (i.e., $\{D_i\}$ and ca_i)

We show the detailed process of computing the dynamic graph structures in Algorithm 1 and then summarize our proposed method in Algorithm 2. In each iteration, we first estimate the graph structure using the PDE and the current value of ca_i and $\{D_i\}$. Then we update ca_i and $\{D_i\}$ as well as other model parameters using available training observations.

5.4 Experimental Results

We evaluate the proposed method for predicting stream temperature using real-world data collected from the Delaware River Basin, which is an ecologically diverse region and a societally important watershed along the east coast of the United States as it provides drinking water to over 15 million people. We first describe our dataset and baselines. Then we discuss the results about the predictive performance using sparse data, the spatial distribution of errors, and model generalization. All experiments are conducted using Tensorflow on

a computer with the following configuration: Intel Core i7-8750H CPU @2.20GHz $\times$ 6 Processor, 16 GiB Memory, 64-bit Win10 OS.

5.4.1 Dataset and Baselines

The dataset is pulled from U.S. Geological Survey's National Water Information System [178] and the Water Quality Portal [155], the largest standardized water quality dataset for inland and coastal waterbodies [155]. Observations at a specific latitude and longitude were matched to river segments that vary in length from 48 to 23,120 meters. The river segments were defined by the national geospatial fabric used for the National Hydrologic Model as described by Regan et al. [157], and the river segments are split up to have roughly a one day water travel time. We match observations to river segments by snapping observations to the nearest stream segment within a tolerance of 250 m.

We study a subset of the Delaware River Basin (Christina River Watershed) with 42 river segments that feed into the mainstem Delaware River at Wilmington, Delaware. We use input features at the daily scale from Oct 01, 1980, to Sep 30, 2016 (13,149 dates). The input features have 10 dimensions which include daily total precipitation, daily mean air temperature, day of the year, solar radiation, shade fraction, potential evapotranspiration and the geometric features of each segment (e.g., elevation, length, slope and width). Water temperature observations were available for 32 segments but the temperature was observed only on certain dates. The number of temperature observations available for each segment ranges from 1 to 9,810 with a total of 51,103 observations across all dates and segments. We compare model performance to multiple baselines, which are described as follows:

- Artificial neural networks (ANN): We train an ANN model using data collected from all the segments on all the dates. The model is applied to predict water temperature on each date separately.
- Recurrent neural networks (RNN): We train an RNN model with the LSTM cell for modeling temperature dynamics across consecutive dates. The RNN model takes the daily input drivers but the loss is only defined on those dates with observations.
- HydroNets [131]: This method also takes into account both the temporal dependencies and spatial river structures using customized model architecture. It has poor performance on our dataset because its river-specific model parameters cannot be effectively trained for segments without observations.
- Recurrent Graph Neural Networks (RGrN) [92]: This baseline combines the graph convolutional networks and LSTM, and has shown promising results in predicting water temperature in streams.

- 2-points DGN: This is a variant of the proposed method, which only utilizes one closest upstream segment when estimating the second-order derivatives (as discussed in Section 5.3.2).

All the models are trained and applied to all the river segments. In the following experiments, we train each ML model using data from the first 24 years (Oct 01, 1980, to Sep 30, 2004) and then test in the next 12 years (Oct 01, 2004, to Sep 30, 2016). The hidden representation in these ML models is in 20 dimensions. We set the learning rate to be 0.0005 and update the model for 100 epochs for modeling water temperature.

5.4.2 Predictive Performance using Sparse Data

We report the testing performance of different methods for temperature prediction and streamflow prediction in Table 5.1. We also test the capacity of each model to learn using less training data by randomly selecting 5% and 10% labeled data from first 24 years for training the model. We repeat each experiment five times with random model initialization and random selection of sparser data (5%, 10%) and report the mean and standard deviation of the root mean square error (RMSE).

We observe that all the methods have larger RMSE values when we reduce the amount of training data. Our proposed method PDE-DGN outperforms baselines using different amounts of training data. In particular, RGrN and HydroNets perform worse than PDE-DGN because these models use a static graph based on the river network structure and thus they are limited in fully capturing dynamic interactions amongst streams. Although the standard graph convolutional structure can also extract different hidden representation for different segments (given their individual input features) and propagate the extracted information to the neighbors, such extraction process is conducted using a set of parameters shared over all the segments and over time. Hence, they cannot model how segment-specific physical variables (e.g., cross-sectional areas) and time-varying variables (e.g., streamflow) affect the segment interactions. Moreover, by leveraging the knowledge encoded by the PDE, the proposed method stands a higher chance at extracting more generalizable patterns. Our method also outperforms the 2-points DGN method, which confirms the effectiveness of incorporating multiple upstream segments in estimating the second-order derivatives for representing the heat diffusion process.

Also, the improvement from RNN to RGrN shows that the incorporation of upstream-downstream dependencies in river networks is helpful to improve the accuracy for predictions. Such improvement is especially obvious as we use less training data. In terms of speed, the overall running time for a complete PDE-DGN model is around 45 minutes which is slightly higher than RGrN method since the only extra overhead during training stage is updating the adjacent matrix.

Table 5.1: Average RMSE (± standard deviation) from five runs for temperature prediction using 5%, 10%, and 100% training labels. Here our method is compared with Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), HydroNets, RGrN, and PDE-DGN. Bolded values indicate the best performing model for each of the percent training labels used.

Method	5%	10%	100%
ANN	3.706±0.114	2.159±0.059	1.529±0.017
RNN	1.841±0.107	1.731±0.119	1.484±0.051
HydroNets	1.768±0.120	1.666±0.021	1.474±0.016
RGrN	1.744±0.073	1.654±0.077	1.443±0.017
2 points DGN	1.756±0.088	1.633±0.021	1.459±0.046
PDE-DGN	**1.740**±0.081	**1.574**±0.063	**1.428**±0.024

5.4.3 Assessing Performance on Unobserved Segments

One important task for modeling river networks is to make predictions on unobserved river segments, which commonly exist in real-world basins. In this test, we evaluate different models for predicting river segments with no observation data. We report the results in Table 5.2. Here Seg A to Seg E are five river segments that have sufficient observation data for stream temperatures. Each row shows the results for an individual experiment where we intentionally remove the temperature observations for a specific segment during the training period (Oct 01, 1980, to Sep 30, 2004). Then we report the prediction performance of RNN, RGrN, and PDE-DGN only on this segment during the testing period (Oct 01, 2004, to Sep 30, 2016) before and after we remove the training data.

We can observe larger errors produced by all these models after we remove training data for a segment. This is expected because different segments may exhibit different patterns and observations when the target segment is not used for training the model. However, we observe that the drop in performance of PDE-DGN is consistently smaller than that of the RNN model and the RGrN model. This confirms that the incorporation of the governing PDE helps the ML model to learn more generalizable patterns. We can also observe that RGrN generally performs better than the RNN model, which demonstrates the effectiveness of the graph structure in propagating relevant information to unobserved segments.

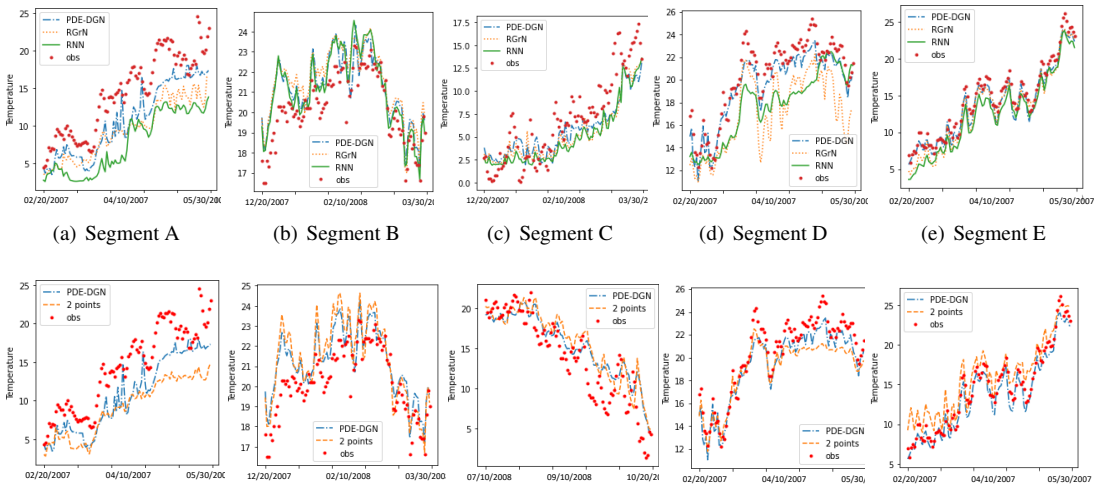
In Fig. 5.4, we show the predictions made by different models on segments A-E after we intentionally hide the training data from each of these segments. PDE-DGN better matches true observations compared with other methods when the model does not have access to the training data from these segments. RNN generally predicts a smoother trajectory but does not capture temperature changes very well.

The 2-points DGN method does not work as well as the complete PDE-DGN because the DGN model only consider one closest upstream. However, if a river segment has a single upstream and downstream segment, the two approaches may perform similarly (e.g., Segment B shown in Fig. 5.4 (g)). In contrast, the segment E has two upstream segments, and that is why PDE-DGN performs better than 2-points DGN.

The segment C also has two upstream segments but the PDE-DGN has similar performance with the 2-points DGN. This is because its two upstreams observation points are far away from the current segment thus making very little contribution (i.e., having much lower weights in the adjacency matrix) to the segment C.

Table 5.2: RMSE of temperature prediction on individual segments after removing training observation data. Here we compare the performance of RNN, RGrN, and PDE-DGN models. Bolded values indicate the best performing model for each segment and training scenario.

Segment	Method	With Obs	Without Obs
Seg A	RNN	2.297±0.082	4.104±0.921
	RGrN	2.176±0.070	3.724±0.637
	PDE-DGN	**2.151**±0.020	**3.127**±0.366
Seg B	RNN	1.116±0.064	1.440±0.068
	RGrN	1.014±0.016	1.387±0.068
	PDE-DGN	**0.994**±0.062	**1.289**±0.055
Seg C	RNN	1.082±0.083	2.302±0.124
	RGrN	1.007±0.032	2.021±0.217
	PDE-DGN	**0.992**±0.027	**1.936**±0.219
Seg D	RNN	0.955±0.053	2.527±0.161
	RGrN	0.943±0.020	2.278±0.391
	PDE-DGN	**0.917**±0.063	**1.971**±0.122
Seg E	RNN	1.067±0.045	1.461±0.097
	RGrN	**0.979**±0.018	1.277±0.063
	PDE-DGN	0.980±0.051	**1.243**±0.046



(a) Segment A (b) Segment B (c) Segment C (d) Segment D (e) Segment E

(f) Segment A (g) Segment B (h) Segment C (i) Segment D (j) Segment E

Figure 5.4: Predictions made by (a-e) RNN, RGrN, PDE-DGN and (f-j) 2-points DGN and the complete PDE-DGN in Segments A-E after we intentionally hide the training data for each segment.

5.4.4 Generalization Test

It is known that traditional ML models are limited in generalizing to a new scenario that is very different from training data. We hypothesize that the proposed PDE-DGN has better generalizability because it follows

general physical relationships that govern underlying processes. Here we test model generalizability for stream temperature prediction across different seasons.

In particular, we train each model using data only from colder seasons (spring, fall and winter) in the first 24 years and then test in summers in the next 12 years, as shown in Table 5.3 (first column). We also show a baseline in which each model is trained using all the data from the first 24 years and then tested in summers in the last 12 years (Table 5.3 second column).

We can observe that PDE-DGN performs better than other methods because of its awareness of the underlying physical relationships. Because the adjacency matrices are created based on the governing PDE, the model has a higher chance to learn physically consistent patterns. Compared to the static graph model of the RGrN, the dynamic graph model of the PDE-DGN is advantageous because the connection strength between stream segments can change substantially in one season compared to another. The HydroNets model performs poorly because it is more likely to overfit the training data due to the higher model complexity.

Table 5.3: Temperature RMSE in summers from 2005 to 2016. Each model is trained using observation data from spring, fall, and winter seasons (Column 1) or the observations data from all the seasons (Column 2) from Oct 1980 to Sep 2004. Here our method is compared against ANN, RNN, HydroNets, RGrN, 2-points DGN, and PDE-DGN). Bolded values indicate the best performing model for each training scenario.

Method	Train on cold seasons	Train on all the data
ANN	2.138±0.093	1.529±0.017
RNN	2.104±0.080	1.484±0.051
HydroNets	2.792±0.018	1.474±0.016
RGrN	2.085±0.046	1.443±0.017
2 points DGN	2.106±0.064	1.459±0.046
PDE-DGN	**2.055**±0.051	**1.428**±0.024

5.5 Conclusion

In this chapter, we propose a novel method called PDE-DGN for modeling interacting segments and predicting temperatures in river networks. We leverage the prior physical knowledge about segment-to-segment interactions embedded in PDE-based models to enhance the learning of latent representation in the proposed ML model. Moreover, we approximate the physical meaning by applying FDMs on the river segments guided by PDEs. Although there were marginal improvements in model performance when trained on all the data, our proposed method demonstrated superiority when handling data-sparse conditions and in generalizing to unseen scenarios. The proposed method also estimates physically meaningful parameters (e.g., PDE coefficients) that could inform other modeling or resource management activities and increase trust in deep learning models. In addition to modeling variables in river networks, the proposed method can be adjusted to model other complex systems which involve dynamic interacting processes.

CHAPTER 6

PDE-Driven Neural Networks for Modeling Temporal Dependencies

6.1 Turbulent Flows Modelling and Reconstruction

Understanding turbulence is the key to our comprehension of many natural and technological processes in engineering, science, and medicine research. Since the pioneering work of Orszag and Patterson ([138]), direct numerical simulation (DNS) of the Navier-Stokes equations have been widely regarded as the computational method with the highest fidelity in capturing the dynamics of turbulent flows. DNS is essentially a brute force numerical solution of the unsteady governing equations of fluid flow, and thus can be very computationally expensive. Straightforward estimates indicate that simulation of an incompressible flow with Reynolds number $\mathrm{Re} = \mathscr{O}(10^5)$ within a domain of size of $\mathscr{O}\left[(100\ell)^3\right]$ would require about *a century* of CPU time on a 1 teraflop computer! A practical alternative, the large eddy simulation (LES) concentrates on the larger scale eddies and models the subgrid-scale transport. By this filtering, LES can be conducted on coarser grids as compared to those required by DNS. The penalty, understandably, is that LES-generated data are of lower accuracy compared to DNS.

Machine learning, especially SR methods ([141]), has shown great success in reconstructing high-resolution data in a variety of commercial applications. The power of these models comes mainly from the use of convolutional network layers ([4]), which can extract the spatial texture features and transform them through complex non-linear mappings to recover high-resolution data. From the earliest end-to-end convolution-based SR model ([53]), many investigators have added skip-connections in SR models ([3, 46, 55, 179, 219, 220]) to bypass redundant low-resolution information and promote the stability of optimizing deep networks. Moreover, advances in adversarial learning allow preservation of high-level features extracted from target high-resolution images through a separate discriminator network ([41, 43, 97, 113, 191, 195, 196, 200]). Given their success in computer vision, SR models have also been applied to reconstruct turbulence data ([51, 65, 66, 120, 137, 176, 192, 210]). However, existing SR methods face several challenges when they are applied for reconstructing turbulent flows. Turbulent flows involve multiple physical variables and often exhibit complex dynamic patterns, i.e., multiple physical variables evolve and interact at different scales. In the absence of underlying physical processes, pure data-driven SR models require a large number of training samples to capture the correct physics. Due to the substantial computational cost in simulating turbulent flows, high-fidelity DNS data are rarely available, and even the generation of high-quality LES at a lower resolution can be expensive. Hence, low-resolution LES data cannot be frequently generated for a variety of simulation

scenarios. When trained with limited data at discrete time steps (i.e., when both LES and DNS are available), these models can have degraded performance because they may learn spurious patterns between sparse observations, and such patterns are often not generalizable.

In this chapter, we propose a new physics-guided neural network framework for spatial and temporal super-resolution. The idea is to leverage underlying physical relationships to guide the learning of generalizable spatial and temporal patterns in the reconstruction process. In particular, our framework consists of two components, physics-guided recurrent unit (PRU) and physics-guided super resolution model (PGSR). The PRU structure is designed based on the underlying PDE, and is responsible for capturing the temporal dynamics of turbulent flows from sparse data. The PGSR model incorporates additional physical constraints to improve the reconstruction from the available LES data. Our evaluation of the Taylor-Green Vortex data ([29]) has demonstrated the superiority of PRU and PGSR in modeling the turbulent flows. We also verify that the proposed method can preserve the physical properties of turbulent flows.

Our contributions can be summarized as:

- We propose innovative physics-guided PRU and PGSR architectures to capture the temporal and spatial patterns of the turbulent flows, respectively.
- We design a unified neural network framework combining PGSR and PRU to effectively simulate and reconstruct high-resolution frequent turbulent flows.
- We evaluate our model in a series of experiments. The experimental results demonstrate that our approaches have significant superiority compared with existing methods in both DNS simulation from historical data and DNS reconstruction from sparse LES data.

6.2 Problem Definition

Our objective is to reconstruct frequent high-resolution flow data from low-resolution and sparse LES data. In particular, we consider a general three-dimensional vortex flow over space and time $Q(x,y,z,t)$, where (x,y,z) denotes the spatial coordinates, t represents the time step (in seconds), and $Q(x,y,z,t)$ consists of multiple variables that describe turbulent transport, such as the velocity along with different directions and the thermodynamic pressure. We represent low-resolution LES data as $Q^{LR}(x,y,z,t)$, which are available at sparse time steps, e.g., starting from a time step t_0, the LES is generated with a time interval of d at $\{t_0, t_0 + d, t_0 + 2d, ...\}$. The flow variables in $Q(x,y,z,t)$ also follow the Navier-Stokes equation, which governs the transport of these variables in space (x,y,z) and time (t). Boundary conditions are specified near the boundary of the domain to describe the interaction of the flow with the external environment. More details about the flow dataset will be provided in Section 6.4.1.

6.3 Method

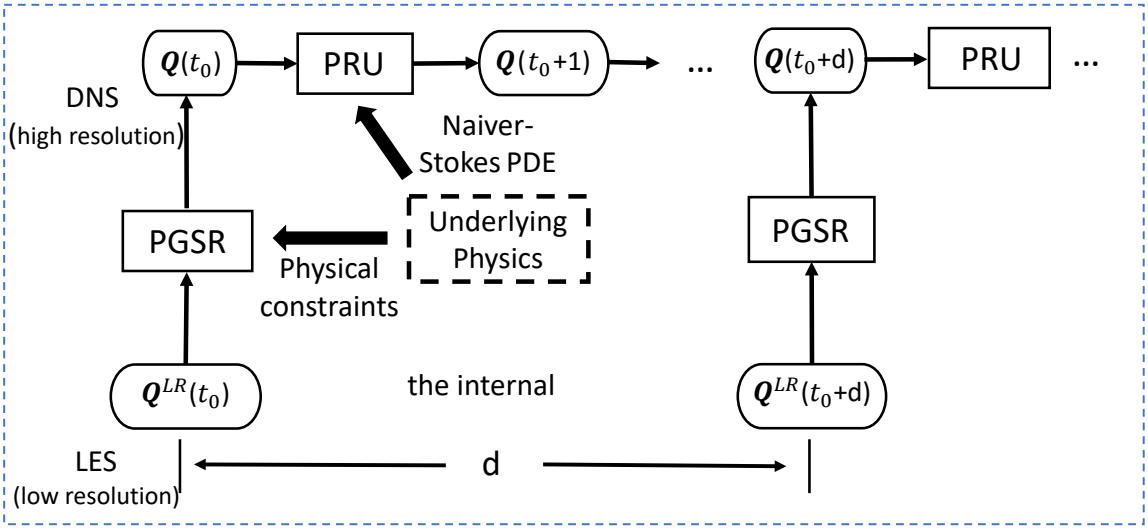


Figure 6.1: The proposed physics-guided neural networks framework combining PRU and PGSR for reconstructing turbulent flows Q.

Our proposed framework consists of two structural components, PGSR and PRU, which are illustrated in Fig. 6.1. Starting from an initial time step t_0, the proposed method will follow a two-step process: (i) the PGSR model is used to reconstruct high-resolution $Q(x,y,z,t)$ when low-resolution LES data are available. (ii) Then PRU is used to estimate $Q(x,y,z,t+1)$ from $Q(x,y,z,t)$ until the next LES sample is available. In the following, we will describe the two components, PRU and PGSR.

6.3.1 Physics-Guided Recurrent Unit (PRU)

Physical variables Q in turbulent flows interact with each other and evolve at different speeds for different locations. Temporal neural network models, e.g., LSTM ([87]), have sophisticated structures and thus heavily rely on large representative training data that are sampled at the high temporal frequency to capture the underlying continuous patterns over time.

Given sparse and limited LES data, we develop the PRU structure as a more accurate and reliable way to predict the future flow variables by leveraging the continuous physical relationship described by the underlying PDE. This helps bridge the gap between discrete data samples and continuous flow dynamics. The proposed PRU structure is also generally applicable to many dynamical systems with governing PDEs.

Most PDEs can be represented in the form of $Q_t = f(t,Q;\theta)$, where Q_t is the temporal derivative of Q, and $f(t,Q;\theta)$ is a non-linear function (parameterized by coefficient θ) that summarizes the current value of Q and its spatial context. For example, the incompressible Navier-Stokes equation for the velocity field can

be expressed as:

$$f(Q) = \frac{-1}{\rho}\nabla p + \nu\Delta Q - (Q.\nabla)Q, \tag{6.1}$$

where ρ, p, and ν denote the fluid density, the thermodynamic pressure, and the viscosity, respectively. Since the function $f(Q)$ in the Navier-Stokes equation is independent of time t, we omit the independent variable t in the function $f(\cdot)$. Here p is treated as a known variable, and $\theta = \{\rho, \nu\}$.

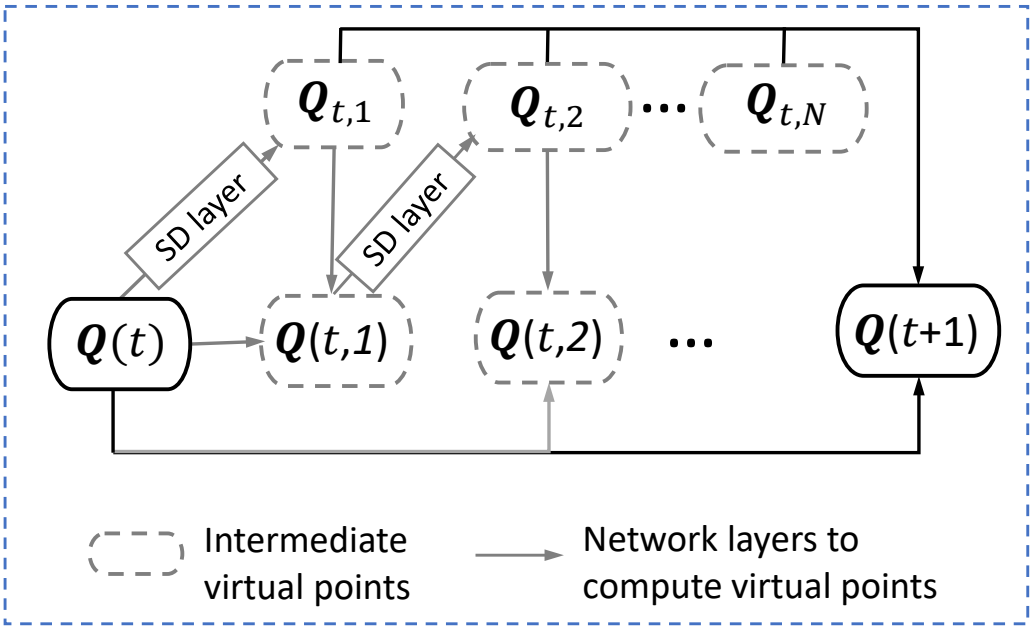


Figure 6.2: Diagram for the physical recurrent unit.

The PRU structure is inspired by the classical numerical Runge–Kutta (RK) methods ([33]), which have been used in temporal discretization for the approximate solutions of differential equations. As shown in Fig. 6.2, the central idea of PRU is to interpolate virtual intermediate variables and create smaller intervals between two time steps which facilitate refining the gradient of flow variables over time. Starting from a time step t, PRU estimates $N-1$ intermediate state variables $Q(t,1)$, ..., $Q(t,N-1)$ and N intermediate temporal derivatives $Q_{t,1}$, ..., $Q_{t,N}$ before reaching the next step $t+1$.

In particular, PRU interpolates intermediate state variables by iteratively following a two-step process: for n from 1 to N, (i) PRU first estimates the temporal derivative $Q_{t,n} = f(Q(t,n-1))$ at the previous intermediate flow state $Q(t,n-1)$, and $Q(t,0) = Q(t)$. We will discuss more details about how to compute the function $f(\cdot)$ later. (ii) Then PRU computes the next intermediate state variable $Q(t,n)$ by moving the flow data $Q(t)$ along the direction of obtained temporal derivatives. In our tests, we follow the most popular 4^{th} order RK

method for computing the three intermediate state variables, as follows:

$$\begin{aligned} Q(t,1) &= Q(t) + \Delta t \frac{Q_{t,1}}{2}, \\ Q(t,2) &= Q(t) + \Delta t \frac{Q_{t,2}}{2}, \\ Q(t,3) &= Q(t) + \Delta t Q_{t,3}, \end{aligned} \tag{6.2}$$

The temporal derivative $Q_{t,4}$ is then computed from the last intermediate point, as $f(Q(t,3))$. The 4^{th} order RK method has the total accumulated error of $O(\Delta t^4)$, where Δt represents the time interval between consecutive time steps.

Finally, PRU combines all the intermediate temporal derivatives as a composite gradient to predict the flow variables at the next time step $Q(t+1)$, as follows:

$$\text{PRU}(\hat{Q}(t+1)|Q(t)) = Q(t) + \sum_{n=1}^{N} w_n Q_{t,n}, \tag{6.3}$$

where $\{w_n\}_{n=1}^{N}$ are the trainable model parameters. Given a series of high-fidelity DNS training data of T time steps, the PRU structure can be trained by minimizing the mean squared error (MSE) between the predicted flow variables and true DNS values, as $\sum_t ||\text{PRU}(\hat{Q}(t+1)|Q(t)) - Q(t+1)||^2/T$.

In the following, we will describe two major issues in computing the function $f(\cdot)$: (i) estimating spatial derivatives in the function $f(\cdot)$, and (ii) preserving boundary conditions. We will also investigate the stability of this method for long-term prediction with a simple case.

6.3.1.1 Spatial Derivative Approximation

The proposed PRU evaluates the function $f(\cdot)$ explicitly for estimating the temporal derivatives of intermediate state variables. In many general PDEs (e.g., the Navier-Stokes equation), $f(Q)$ contain spatial derivatives of Q. One popular approach for evaluating the spatial derivatives is through the finite difference methods (FDMs), which approximate variable derivatives of a function on predefined mesh points by solving algebraic equations containing finite differences and values from nearby points. In particular, the first and second order spatial derivatives (along x dimension) in Eq. 6.1 can be estimated by FDMs as follows:

$$\begin{aligned} Q_x(x_i, y_j, z_k, t_n) &\approx \frac{Q(x_{i+1}, y_j, z_k, t_n) - Q(x_{i-1}, y_j, z_k, t_n)}{2\Delta x}, \\ Q_{xx}(x_i, y_j, z_k, t_n) &\approx \frac{Q(x_{i+1}, y_j, z_k, t_n) - 2Q(x_i, y_j, z_k, t_n)}{2\Delta x} \\ &\quad + \frac{Q(x_{i-1}, y_j, z_k, t_n)}{2\Delta x}. \end{aligned} \tag{6.4}$$

The approximation used in FDMs results in an error compared to the exact solution, which can be esti-

mated through Taylor expansions. Instead of using FDMs for every mesh point, we propose to build a spatial difference (SD) layer using CNN layers. The CNN layers have the expressive power to capture the relationships defined in FDMs (Eq. 6.4) while also being more flexible in learning other non-linear relationships from data.

6.3.1.2 Boundary Condition and Augmentation

Boundary conditions are critical in turbulent flow simulation as they describe how the turbulent flows interact with the external environment. Here we consider the periodic boundary condition in our flow data. It is defined in a specified periodic domain indicating that it repeats its own values in all directions. The formal definition of a cubic periodic boundary condition is given below:

$$\begin{aligned} Q(L_x, y, z, t) &= Q(R_x, y, z, t), \\ Q(x, L_y, z, t) &= Q(x, R_y, z, t), \\ Q(x, y, R_z, t) &= Q(x, y, R_z, t), \end{aligned} \tag{6.5}$$

where L_x, L_y, L_z are the three left boundaries with respect with x, y, z coordinates and R_x, R_y, R_z are the three right boundaries with respect with x, y, z coordinates. Standard padding strategies for CNN (e.g., same padding) do not satisfy the periodic value requirement. In order to handle this issue, we make a data augmentation for each of the 6 faces (of the 3D cubic data) with an additional 2 layers of data during the training stage and adopt a 5×5 CNN filter size. The augmented locations will be removed from reconstructed data.

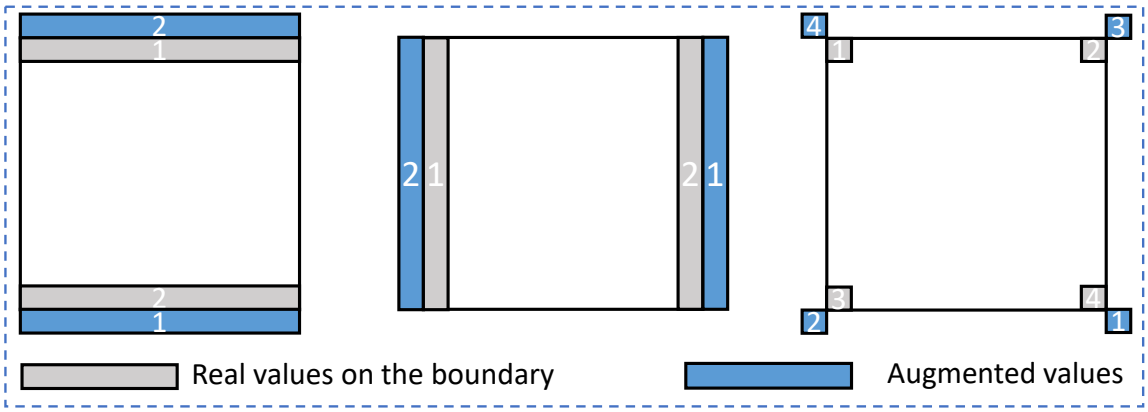


Figure 6.3: Illustration of data augmentation on a 2-D example. The left diagram represents the up and low boundary augmentation. The middle diagram represents the left and right boundary augmentation. The right diagram represents the corner boundary augmentation. Rectangles carrying identical numbers have the same value.

6.3.1.3 Stability

The classical 4^{th} order RK suffers from the stability issue if the step size is not properly chosen. Consider a simple scalar example $Q_t = \lambda Q$. The 4^{th} order RK for this equation can be written as

$$Q((n+1)\Delta t) \approx (1+\lambda\Delta t + \frac{\lambda\Delta t^2}{2} + \frac{\lambda\Delta t^3}{6} + \frac{\lambda\Delta t^4}{24})Q(n\Delta t). \tag{6.6}$$

Let's denote $R(\Delta t)$=$1+\Delta t + \frac{\Delta t^2}{2} + \frac{\Delta t^3}{6} + \frac{\Delta t^4}{24}$, and we have $Q((n+1)\Delta t) = R(\Delta t)Q(n\Delta t)$. The analytical solution is $Q((n+1)\Delta t) = \exp(\lambda\Delta t)Q(n\Delta t)$, and thus the accumulated error is

$$err_{n+1} = (\exp(\lambda\Delta t) - R(\Delta t))err_n. \tag{6.7}$$

This indicates that $err_{n+1} = O(\Delta t^5)err_n$ according to Taylor expansion. When the interval d of LES data is large, the accumulated error may get amplified at every time step and then lead to an explosion. Additional complexity arises when f consists of multiple evaluations of spatial derivatives. This requires the access to LES data at a reasonably frequent time interval to avoid significantly large reconstruction errors.

Algorithm 5 The flow of the proposed PRU.

```
Create and initialize 5 × 5 filters for 1st and 2nd order spatial derivatives
for epoch = 1 : number of training iterations do
    for t = 1 : number of time steps do
        Make data augmentation for Q(t) (Section 6.3.1.2).
        Calculate Q_{t,1}, Q_{t,2}, Q_{t,3}, Q_{t,4} following Eq. 6.2 and evaluate f accordingly.
        Calculate Q̂(t+1) following Eq. 6.3 and remove augmented data over boundaries.
        Use the predicted Q̂(t+1) as the input flow data for time t+1.
    Update trainable filters and weights.
```

6.3.2 Physics Guided Super Resolution (PGSR)

The PGSR model aims to incorporate additional physical constraints to regularize the standard super-resolution model. In particular, we consider two important physical constraints, the divergence-free property for the incompressible flow and the zero-mean property for the Taylor-Green Vortex ([29]).

First, the incompressible flow follows the divergence-free property in the velocity field. Thus, we can represent the inherent physical relationship of the velocity field as:

$$\nabla \cdot \mathbf{V} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{6.8}$$

where we represent the velocity vector $\mathbf{V}(\mathbf{x},t)$ along 3-D dimensions ($\mathbf{x} \equiv x,y,z$) by u, v, and w, respectively. Then we use a second-order central finite difference approximation to estimate the partial derivatives and

employ this divergent free property as a physical loss in the training process, as follows:

$$\mathscr{L}_{\text{Phy}} = \sum_{(x,y,z)} \left[\nabla \cdot \hat{\mathbf{V}}(\mathbf{x},t)\right]^2 / M, \tag{6.9}$$

where M is the number of spatial locations in the high-resolution data, and $\hat{\mathbf{V}}$ represents the reconstructed velocity field at high resolution. Such physical constraint can help reduce the search space for model parameters such that the reconstructed high-resolution data follow the divergence-free property which is enforced in incompressible flows.

Second, to preserve the zero-mean property of the in a compressible flow, we also implement an extra network layer by reducing the mean value of reconstructed flows in the generative process $\hat{Q}_0 = g(\hat{Q})$. We do not include the zero-mean constraint directly in the loss function because the obtained model cannot preserve the zero-mean property for the long-term testing phase. On the other hand, direct MSE minimization using $\hat{Q}_0$ as output leads to an unstable training process because the original output $\hat{Q}$ can have arbitrarily large values. Hence, we iteratively train the PGSR model to reduce (i) the gap between $\hat{Q}$ and the true DNS and (ii) the gap between $\hat{Q}$ and its resulted $\hat{Q}_0$, and finally use $\hat{Q}_0$ as the output.

Additionally, we also introduce a degradation process to enforce the consistency between the reconstructed data and the input LES data, similar to [38]. We create the PGSR model based on the popular SR model SRGAN ([113]). The methods we used to include physical constraints can also be used to enhance other SR models.

6.4 Experiment

In this section, we evaluate the performance of our method on a Taylor-Green vortex (TGV) ([29]) dataset and compare the results with existing well-used methods. We first introduce the dataset used in our tests and discuss the experimental design and evaluation targets. Then we will provide experimental results and our analysis.

6.4.1 Dataset

We consider a variant of the TGV. This is a three-dimensional incompressible flow and is simulated within a box with periodic boundary conditions. The TGV provides a suitable setting for our demonstration as it exhibits several salient features of turbulent transport. In this flow, the original vortex collapses into turbulent worm-like structures which become progressively more turbulent until viscosity eventually dissipates the large scale vortical structures. We compare our proposed method against several existing super-resolution algorithms to reconstruct the DNS data of TGV.

The TGV is produced by a solution of the constant density Navier-Stokes equation:

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V}.\nabla)\mathbf{V} = \frac{-1}{\rho}\nabla p + \nu\Delta\mathbf{V}. \tag{6.10}$$

The evolution of the TGV includes enhancement of vorticity stretching and the consequent production of small-scale eddies. Initially, large vortices are placed in a cubic periodic domain of $[-\pi, \pi]$ (in all three-directions), with initial conditions:

$$u(x,y,z,0) = \sin(x)\cos(y)\cos(z) \tag{6.11}$$

$$v(x,y,z,t) = -\cos(x)\sin(y)\cos(z) \tag{6.12}$$

$$w(x,y,z,t) = 0. \tag{6.13}$$

Then the value of the Reynolds number is set to $Re = 1600$. We have LES and DNS results of TGV at several times steps. For each time step, we consider the three components of the velocity along the x, y, and z axis, denoted by u, v, and w, respectively. Our objective is to reconstruct the DNS results of the velocity field (u, v, w) using LES data. In particular, Q^{LR} represents the LES values of the velocity field while the target Q represents the high-fidelity DNS of the velocity field. Here both LES and DNS data are generated along 65 grid points along the z axis under equal intervals. The LES and DNS are conducted on 32-by-32 and 128-by-128 grid points, respectively, along the xy directions. Hence, the DNS data is of 16 times higher resolution compared to LES data.

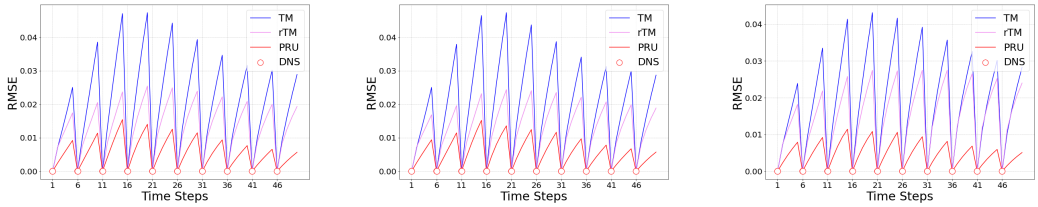
6.4.2 Experimental Design

We train the proposed method using the TGV data from a consecutive 20-seconds period (with 20 time steps) and then apply the trained model to the next 50 seconds' testing data and measure the performance. [1] We evaluate the performance of DNS prediction using two different evaluation metrics, root mean squared error (RMSE) and structural similarity index measure (SSIM) ([197]). We use RMSE to measure the difference (error) between reconstructed data and target DNS data. The lower value of RMSE indicates better reconstruction performance at the pixel level. SSIM is used to appraise the structural similarity between reconstructed data and target DNS on three aspects: luminance, contrast, and overall structure.

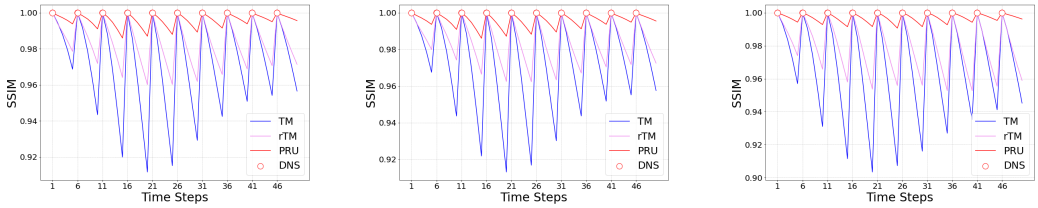
Our evaluations aim to answer several questions as listed below:

E1: *Whether PRU alone can effectively predict the next high-resolution DNS using the previous DNS data?* We will compare PRU with two pure data-driven baseline models, transition model (TM) and recurrent

[1] Code for the experiment is available at drive.google.com/drive/folders/11PTaEjsBkgd6PAAYmWH_KDzTg90IvrJn?usp=sharing

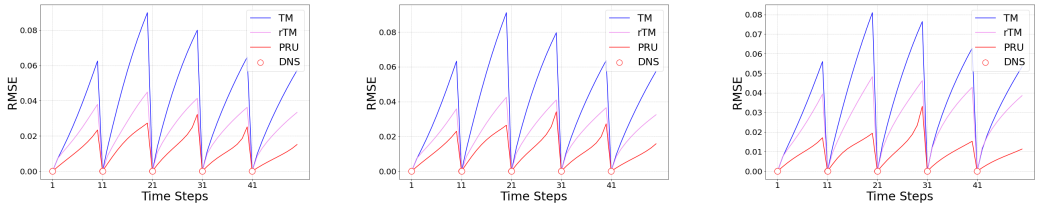


(a) RMSE in u Channel. (b) RMSE in v Channel. (c) RMSE in w Channel.

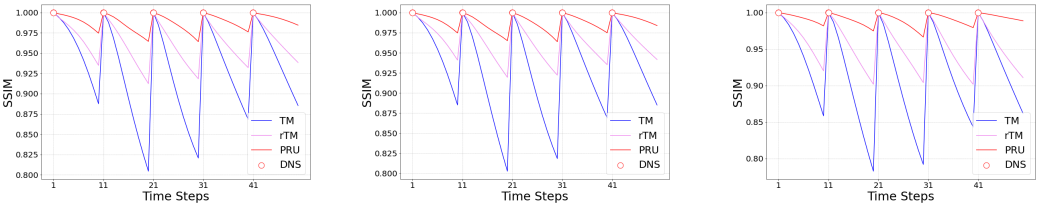


(d) SSIM in u Channel. (e) SSIM in v Channel. (f) SSIM in w Channel.

Figure 6.4: Change of RMSE/SSIM values produced by different DNS prediction models from the 1st to 50th time steps in a testing period with true DNS data for 5 time steps. (a)-(c) show the changes of RMSE values, and (d)-(f) show the changes of SSIM values for (u,v,w) three different channels.



(a) RMSE in u Channel. (b) RMSE in v Channel. (c) RMSE in w Channel.



(d) SSIM in u Channel. (e) SSIM in v Channel. (f) SSIM in w Channel.

Figure 6.5: Change of RMSE/SSIM values produced by different DNS prediction models from the 1st to 50th time steps in a testing period with true DNS data for 10 time steps. (a)-(c) show the changes of RMSE values, and (d)-(f) show the changes of SSIM values for (u,v,w) three different channels.

transition model (rTM). The TM method predicts the flow variables $Q(t+1)$ at next step using an UNet-style encoder-decoder convolutional structure from the flow variables $Q(t)$ at the previous time. The rTM method further extends TM with a recurrent layer.

E2: *Whether the predictions made by PRU can preserve physical properties of DNS?* Besides RMSE and

SSIM, we will measure the turbulent kinetic energy of the predicted flows and compare it with that of the true DNS.

E3: *How is the reconstruction performance combining PRU and PGSR using sparse low-resolution LES data?* We will combine PRU and PGSR (PGSR-PRU) for reconstructing DNS from sparse LES samples. Since we have already compared PRU with other temporal transition models in **E1**, here we compare to a baseline SRGAN-TM, which uses our base SR model SRGAN for reconstructing DNS from LES and use TM to predict DNS when LES is not available. We also compare to another two baselines PGSR (LES) and its extension rPGSR (LES). rPGSR(LES) has another recurrent layer over time. Different from PGSR-PRU and SRGAN-TM, these two methods apply the SR model using LES data at all the time steps, thus can be considered as the upper bound for this test. Our goal is to verify that PGSR-PRU can produce comparable performance with PGSR (LES) and rPGSR (LES).

E4: *How is the reconstruction performance of PGSR compared to other SR methods?* We compare PGSR with two well-used SR methods: RCAN ([219]) and SRGAN ([113]). We also compare it with DC-S/MS ([65]), which is a popular SR approach for turbulent flows reconstruction. Additionally, we compare to a variant of PGSR, termed PGSR-D, which only adds the degradation loss to the SRGAN model without using any physical constraints.

Table 6.1: DNS prediction performance by RMSE and SSIM. The performance is measured on (u, v, w) channels with DNS interval d as 5 or 10.

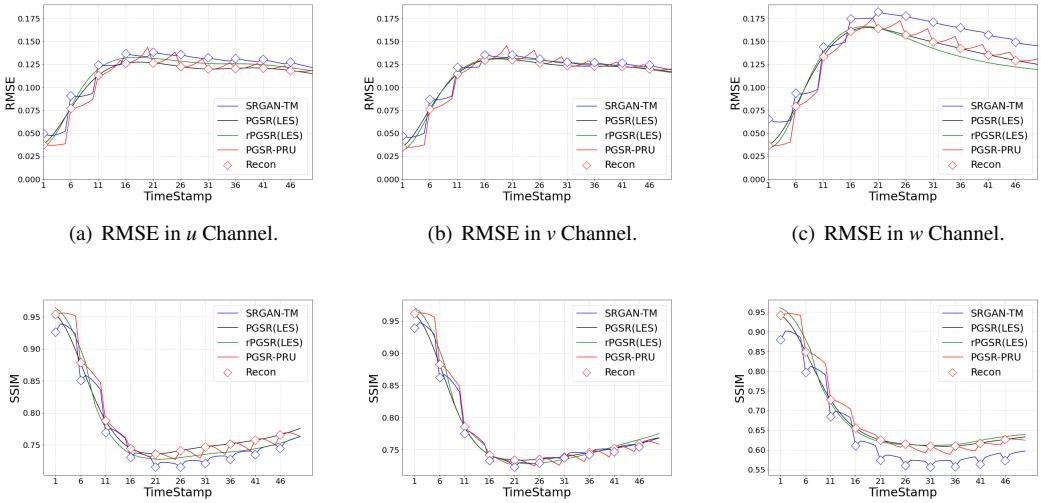
	Method	RMSE	SSIM
	TM	(0.019,0.019,0.019)	(0.972,0.973,0.967)
d	rTM	(0.013,0.012,0.015)	(0.983,0.984,0.980)
=5	PRU	(0.005,0.005,0.005)	(0.996,0.996,0.997)
	TM	(0.038,0.038,0.036)	(0.930,0.930,0.917)
d	rTM	(0.022,0.022,0.025)	(0.964,0.966,0.954)
=10	PRU	(0.012,0.012,0.009)	(0.988,0.988,0.991)

6.5 Results

6.5.1 DNS Generation using PRU

Here we assume that we have true DNS data with an interval of d time steps (d = 5 or 10) and we implement PRU and other baselines to predict DNS for the missing time steps. We summarize the performance of PRU and baselines in Table 6.1 and show their performance change on each channel over time in Figs. 6.4 and 6.5. For both cases (with the true DNS interval d set to a larger value 10 or a smaller value 5), PRU produces better performance than baselines over all the time steps. It confirms the effectiveness of PRU in the long-term prediction of DNS from historical flow data (**E1**).

Besides, we compute the kinetic energy of the flow data predicted by PRU and baselines and measure the



(a) RMSE in u Channel. (b) RMSE in v Channel. (c) RMSE in w Channel.

(d) SSIM in u Channel. (e) SSIM in v Channel. (f) SSIM in w Channel.

Figure 6.6: Change of RMSE/SSIM for different models over time using sparse LES data with the interval of 5 time steps. (a)-(c) show the changes of RMSE values, and (d)-(f) show the changes of SSIM values for (u,v,w) three different channels.

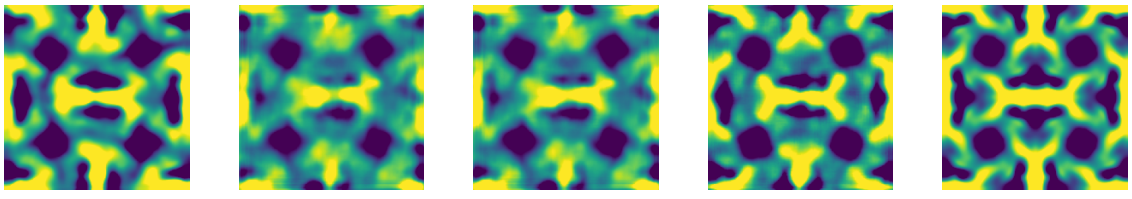
gap with the kinetic energy of the true DNS data. The proposed PRU reduces the kinetic energy gap with the true DNS by 30% and 67% compared to TM and rTM, respectively. It confirms that PRU can better preserve underlying physical characteristics of turbulent flows (**E2**).

Table 6.2: Reconstruction performance on (u, v, w) using LES channels by RMSE and SSIM. SRGAN-TM and PGSR-PRU (proposed) are evaluated using sparse LES data with the interval of 5 steps, the upper half is the average results of a total of 50 time steps, the bottom half is the average results of the first 15 time steps.

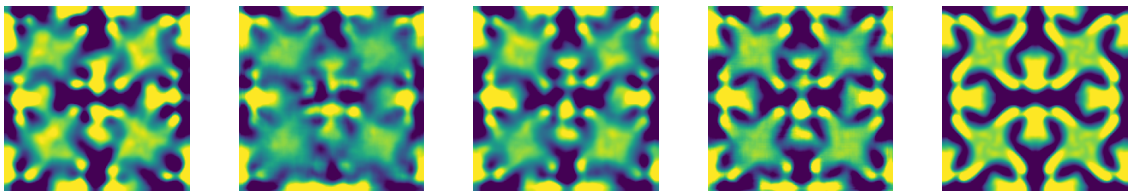
Method	RMSE	SSIM
PGSR (LES)	(0.112, 0.114, 0.133)	(0.771, 0.774, 0.667)
rPGSR (LES)	(0.114, 0.114, 0.129)	(0.772, 0.773, 0.669)
SRGAN-TM	(0.118,0.115, 0.147)	(0.769, 0.767, 0.647)
PGSR-PRU	(0.111,0.113, 0.128)	(0.782, 0.781, 0.681)
PGSR (LES)	(0.088, 0.088, 0.101)	(0.846, 0.849, 0.801)
rPGSR (LES)	(0.091, 0.086, 0.099)	(0.848, 0.855, 0.811)
SRGAN-TM	(0.091,0.088, 0.105)	(0.848, 0.849, 0.794)
PGSR-PRU	(0.081,0.081, 0.091)	(0.864, 0.866, 0.833)

6.5.2 DNS Reconstruction using PGSR-PRU

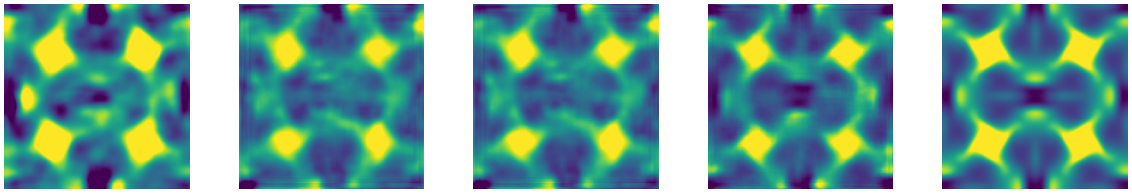
We implement the DNS reconstruction using PGSR-PRU and SRGAN-TM using the LES data for every five time steps (**E3**). As shown in Table 6.2 and Fig. 6.6, PGSR-PRU produces better performance than SRGAN-TM. Particularly in the first 15 time steps, it is more clear to see PGSRN-PRU can obtain lower RMSE and higher SSIM values. Fig. 6.6 also shows that the reconstruction performance gets degraded over time



(a) SRGAN-TM (b) PGSR(LES) (c) rPGSR(LES) (d) PGSR-PRU (e) Target DNS



(f) SRGAN-TM (g) PGSR(LES) (h) rPGSR(LES) (i) PGSR-PRU (j) Target DNS.



(k) SRGAN-TM (l) PGSR(LES) (m) rPGSR(LES) (n) PGSR-PRU (o) Target DNS

Figure 6.7: Three example slides of reconstructed w channel (in three rows) along the z dimension with the LES interval of 5 time steps.

Table 6.3: Evaluation of SR models in terms of the reconstruction RMSE and SSIM on (u, v, w) channels using LES data. The performance is measured on the testing data of the first 5 time steps.

Method	RMSE	SSIM
RCAN	(0.061, 0.061, 0.075)	(0.891, 0.891, 0.863)
DCS/MS	(0.085, 0.086, 0.115)	(0.896, 0.897, 0.845)
SRGAN	(0.065, 0.062, 0.067)	(0.901, 0.913, 0.875)
PGSR-D	(0.057, 0.052, 0.053)	(0.914, 0.923, 0.900)
PGSR	(0.053, 0.050, 0.051))	(0.924, 0.935, 0.911)

because the LES data have a significant difference with the training period. More interestingly, we notice that PGSR-PRU even outperforms PGSR (LES) and rPGSR (LES). This is because LES data often miss many important physical components compared to the true DNS, which makes SR models difficult to recover flow data directly from LES data. We also show three sets of examples of reconstructed slides of flow data in Fig. 6.7. It is clear to observe that PGSR-PRU can better capture the detailed flow patterns compared to other methods as it incorporates the underlying Navier-Stokes equation through PRU.

6.5.3 DNS Reconstruction using PGSR

As shown in Table 6.3, PGSR achieves better performance than other baselines in terms of both RMSE and SSIM. In particular, we can observe the improvement from SRGAN to PGSR-D and from PGSR-D to PGSR.

This confirms the effectiveness of the degradation process and the physical constraints used in PGSR (**E4**).

6.6 Conclusion

In this chapter, we develop a physics-guided neural network framework for predicting high-resolution flow data at high temporal frequency. The PRU structure leverages the physical knowledge embodied in the Navier-Stokes equation to capture the flow dynamics over time, while the PGSR model incorporates additional physical constraints to improve the reconstruction from the LES data. We have demonstrated the superiority of PRU in predicting future DNS data from historical DNS data. We also show that PGSR-PRU can effectively reconstruct DNS from sparse LES series.

More importantly, the proposed method is generally applicable to many scientific problems with similar properties, e.g., complex temporal dynamics, and the availability of low-resolution simulations with reduced accuracy. The PRU structure can also be used as a building block to enhance existing deep learning models for modeling of complex dynamics with the guidance of known governing PDEs.

CHAPTER 7

Transfer Learning using Residual Correction for Inaccurate Physics Laws

7.1 Streamflow Prediction for Multiple Basins

A key problem in Earth science is to build models for simulating heterogeneous processes across large regions, which is also called the regional modelling problem. The central challenge is about how to extrapolate to out-of-sample scenarios, e.g., predicting hydrological flows in ungauged watersheds, from instrumented to non-instrumented hillslopes, from areas with flux towers to areas without, etc. [25]. Often this is done using ancillary data (e.g. soil maps, remote sensing, digital elevation maps, etc.) to help understand similarities and differences between different areas. The regional modelling problem is thus closely related to the problem of prediction in ungauged basins [26, 174]. This problem is well-documented in several review papers, therefore we point the interested reader to the comprehensive reviews by [88, 154], and to the more recent review in the introduction by [145].

This work is focused on predicting streamflow in river basins, which is an important problem in hydrology as it helps ensure water supply, provides early warnings for flooding and droughts, and aids in a better understanding and management of aquatic ecosystems. Traditionally, process-based models have been built for streamflow prediction, and they are often calibrated separately for each river basin.

Currently, the most successful hydrological models are calibrated to one specific basin. These models aim to derive hydrologic parameters that can be used to run simulation models from available data (i.e., observable catchment characteristics). Additional complexity arises due to strong interaction between individual model parameters (e.g., between soil porosity and soil depth, or between saturated conductivity and an infiltration rate parameter), such that any meaningful joint probability distribution over model parameters will be complex and multi-modal. This is closely related to the problem of equifinality [23].

Model-dependent regionalization has enjoyed major attention from the hydrological community, so today a large variety of approaches exist. To give a few selective examples, [166] calibrated a conceptual model for 11 catchments and regressed them against the available catchment characteristics. [163] proposed a multiscale parameter regionalization (MPR) method, which simultaneously sets up the model and a regionalization scheme by regressing the global parameters of a set of a-priori defined transfer functions that map from ancillary data like soil properties to hydrological model parameters. [20] calibrated a conceptual model for 1,787 catchments around the globe and used these as a catalog of 'donor catchments', and then extended this library to new catchments by identifying the ten most similar catchments from the library in terms of climatic

and physiographic characteristics to parameterize a simulation ensemble. [145] first regionalized hydrologic signatures [78] using a regression model (random forests), and then calibrated a rainfall-runoff model to the regionalized hydrologic signatures.

With a recent growing interest in data-driven/machine learning-based streamflow modelling, researchers have found that ML models benefit from learning jointly from a collection of different basins [103]. This requires the ML model to learn and encode the difference in hydrologic behaviors across different basins. In this work, we propose a physics-guided meta-transfer learning approach that integrates the behaviors of multiple basins. The main contribution of this work can be summarized as: 1) We utilize a physical equation, which reduces the size of parameters searching space, to help us build the machine learning model. The ML model parameters can converge to the optimal values more quickly with the assistance of the physical model. 2) Even though existing approaches using physics in ML are based on known physical laws that govern relationships between input and output variables (e.g., mass and energy conservation laws), most physics-based models are necessarily approximations of reality due to incomplete knowledge of certain processes or omission of processes to maintain computational efficiency. In order to handle the issue, we use a lasso regression to approximate the residual caused by the inaccuracy of the physical law. In unlabeled datasets, we use a forward neural network to approximate the lasso regression coefficients with the static feature of basins as its input. 3) We make corrections for both labelled and unlabeled datasets using approximated residual and let the corrected labels rejoin the training process as an extra feature of the original LSTM model. In the other approach, we iteratively update the corrected results based on the new predicted output. The training process switches between the labelled and unlabeled datasets.

7.2 Related Work

Data-driven methods learn how to predict streamflow from weather drivers and catchment physical descriptors directly without involving any hydrological process descriptions. Depending on either one or multiple catchments of data used, the data driven model will learn localized or regionalized hydrologic behaviors, respectively. A local model is referred to as the model using hydrologic data from only one catchment. By contrast, when the hydrology data from multiple catchments are used and those catchments cover a wide range of all available hydrologic behaviors, the model is called a global model.

For data-driven methods, one family is the neural network [22, 89]. In recent years, the Long short-term memory networks [87], one subfamily of neural networks, have shown burgeoning applicability in streamflow prediction tasks [102]. LSTM-based methods predict streamflow from antecedent weather drivers. [103] have shown that using physical descriptors will train a universal global LSTM-based model that outperforms process-based individual models given the same forcing data.

Physics-based models of dynamical systems are often used to study engineering [16] and environmental systems [17]. Despite their extensive use, these models have several well-known limitations due to incomplete or inaccurate representations of the physical processes being modelled. Given rapid data growth due to advances in sensor technologies, there is a tremendous opportunity to systematically advance modelling in these domains by using machine learning methods.

In science and engineering applications, a physical model often predicts the values of many variables. Machine learning models can also generate predictions for many variables (e.g., by having multiple nodes in the output layer of a neural network). Most ML algorithms make use of a loss function from an equation that captures the difference between predicted and actual (i.e., observed values) to guide the search for parameter values that attempts to minimize this loss function. For example, for the lake temperature monitoring application, accuracy at the surface and at high depth can be more important than error at the middle levels of the lake.

Residuals are differences between the one-step-predicted output from the model and the measured output from the validation data set. Thus, residuals represent the portion of the validation data not explained by the model. In order to discover the hidden information inside the residual, a few approaches are given below. Symbolic regression [142] and discovery equations from datasets [28, 36, 162] provide alternative ways to find hidden equations from datasets. Symbolic regression outputs the combination of multiplication, division, addition and subtraction of variables which turns out to be very similar to lasso regression in our experiment. Multiplication and division of two features are not easily interpretable. Discovery equations from datasets is not directly related to our problem, since we do not find an underline differential equation that can build a coherent physical relation between features and labels. [28] talks about finding the coefficients of a partial differential equation given the basic form but, in our case, the dataset does not preserve a spatial relation between features and labels which do not guarantee the discovery of a partial differential equation containing both temporal and spatial derivatives.

7.3 Problem Definition

We consider N observed basins and M unobserved basins in a region. For each basin i, we are provided with input features over T daily time step $\mathbf{X}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, ..., \mathbf{x}_i^T\}$. Here input features $\mathbf{x}_i^t$ form a D_x- dimensional vector, which includes weather drivers (e.g., air temperature, precipitation, evapotranspiration) and static features of the basin. Additionally, we have observed water property (e.g., the amount of streamflow) $\mathbf{Y} = \{y_i^t\}$ for certain basins and on certain dates. Our objective is to predict streamflow over all the N basins in the region at a daily scale by leveraging spatial and temporal contextual information.

7.4 Methods

In this section, we formally describe our proposed method, as outlined in Fig.7.1 and 7.2. We first introduce the LSTM as a classical method towards this type of problem. Then we discuss the regularization with residual correction and input augmentation strategy to refine the LSTM model to different basins based on their physical characteristics. Later, we develop another switch training method to reduce the residual and let the residual rejoin the training process.

7.4.1 Preliminaries

The RNN model has been widely used to model temporal patterns in sequential data. The RNN model defines transition relationships for the extracted hidden representation through a recurrent cell structure. In this work, we adopt LSTM to build the recurrent layer for capturing long-term dependencies. LSTM is a special type of recurrent neural network, well suited for the task of rainfall-runoff, basin modelling and most popular and widely used in hydrology for predicting streamflow [39, 117]. The LSTM cell combines the input features $\mathbf{x}^t$ at each time step and the inherited information from previous time steps. Here we omit the subscript i as we do not target a specific river segment.

Each LSTM cell has a cell state $\mathbf{c}^t$, which serves as a memory and allows for preserving information from the past. Specifically, the LSTM first generates a candidate cell state $\bar{\mathbf{c}}^t$ by combining $\mathbf{x}^t$ and the hidden representation at the previous time step $\mathbf{h}^{t-1}$, as follows:

$$\bar{\mathbf{c}}^t = \tanh(\mathbf{W}_c^h \mathbf{h}^{t-1} + \mathbf{W}_c^x \mathbf{x}^t + \mathbf{b}_c). \tag{7.1}$$

where $\mathbf{W}$ and $\mathbf{b}$ are matrices and vectors, respectively, of learnable model parameters. Then the LSTM generates a forget gate f^t, an input gate g^t, and an output gate o^t via sigmoid function $\sigma(\cdot)$, as follows:

$$\begin{aligned} \mathbf{f}^t &= \sigma(\mathbf{W}_f^h \mathbf{h}^{t-1} + \mathbf{W}_f^x \mathbf{x}^t + \mathbf{b}_f), \\ \mathbf{g}^t &= \sigma(\mathbf{W}_g^h \mathbf{h}^{t-1} + \mathbf{W}_g^x \mathbf{x}^t + \mathbf{b}_g), \\ \mathbf{o}^t &= \sigma(\mathbf{W}_o^h \mathbf{h}^{t-1} + \mathbf{W}_o^x \mathbf{x}^t + \mathbf{b}_o). \end{aligned} \tag{7.2}$$

The forget gate is used to filter the information inherited from $\mathbf{c}^{t-1}$, and the input gate is used to filter the candidate cell state at t. Then we compute the new cell state as follows:

$$\mathbf{c}^t = \mathbf{f}^t \otimes \mathbf{c}^{t-1} + \mathbf{g}^t \otimes \bar{\mathbf{c}}^t, \tag{7.3}$$

where $\otimes$ denotes the entry-wise product.

Once obtaining the cell state, we can compute the hidden representation by filtering the cell state using the output gate, as follows:

$$\mathbf{h}^t = \mathbf{o}^t \otimes \tanh(\mathbf{c}^t). \tag{7.4}$$

According to the above equations, we can observe that the computation of $\mathbf{h}^t$ combines the information at the current time step ($\mathbf{x}^t$) and the previous time step ($\mathbf{h}^{t-1}$ and $\mathbf{c}^{t-1}$), and thus encodes the temporal patterns learned from the data.

The final output y is a linear transformation of $\mathbf{h}$. The regular loss function is defined as

$$loss = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} (y_i^t - \hat{y}_i^t)^2 \tag{7.5}$$

where $\hat{y}_i^t$ represents the output of the neural networks at i^{th} basin and t^{th} day.

In section 7.4.3, we use the lasso linear regression to approximate the residual generated by a rough physical law. The basic form of a lasso is given below.

Consider a sample consisting of T cases, each consisting of p input features and a single outcome. Let res_i^t be the output for the i basin at time t and $\mathbf{s}_i^t = (\mathbf{x}_i^t, y_i^t)$ be the input which contains the feature vector and label. The objective of lasso is to solve

$$\begin{aligned} &min \sum_{t=1}^{T} (res_i^t - \beta_{i,0} - \mathbf{s}_i^t \cdot \beta_i)^2 \\ &subject\ to \sum_{l=1}^{p} |\beta_{i,l}| \leq k. \end{aligned} \tag{7.6}$$

Here β_i is the constant coefficient for basin i, $\beta_i = (\beta_{i,1}, \beta_{i,2}, ..., \beta_{i,p})$ is the coefficient vector, and k is a prespecified free parameter that determines the degree of regularization.

7.4.2 LSTM + Regularization on Pseudo Label

We estimate the pseudo labels of streamflow via the following physical equation.

$$max(0, rainfall_i^t - et_i^t - (sw_i^t - sw_i^{t-1})) = q_i^t \tag{7.7}$$

where $rainfall_i^t$ represents daily average precipitation for basin i at time t, et_i^t represents evapotranspiration, sw_i^t represents the soil water for basin i at time t. q_i^t represents the estimated streamflow and is not an accurate prediction. The absolute value on the left-hand side forces the q_i^t to be a positive number. This is because evapotranspiration and soil water condition at each time are estimated by an uncalibrated process-based model, SWAT [135], and are not directly measured through experiments.

It is noteworthy that the pseudo labels can be generated for basins and time periods without streamflow observations. We can modify the training objective (Eq. 7.5) by enforcing the consistency with physical laws via an additional physics-based regularization term. By adding a regularization term, we can avoid model overfitting to the noisy measurement and stay consistent with any physical law. We can control the strength of regularization by the value of the coefficient.

$$loss = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} (y_i^t - \hat{y}_i^t)^2 + \frac{\lambda}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} (q_i^t - \hat{y}_i^t)^2 \tag{7.8}$$

where λ is a predefined regularization coefficient.

Eq. 7.8 takes account of q_i^t associated with y_i^t. An additional benefit of the physics-based regularization term in Eq. 7.8 is that the computation of the regularization does not require true labels. Hence, instead of using the pseudo label in the labelled dataset only, we also incorporate the pseudo label in the unlabeled dataset. M is the number of unobserved basins.

$$loss = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} (y_i^t - \hat{y}_i^t)^2 + \frac{\lambda}{(N+M) \cdot T} \sum_{j=1}^{N+M} \sum_{t=1}^{T} (q_j^t - \hat{y}_j^t)^2 \tag{7.9}$$

7.4.3 Residual Correction

One limitation of the regularization method is that the pseudo labels q_i^t obtained through Eq. 7.7 may not always be accurate because of the bias of process-based models in estimating evapotranspiration and soil moisture. As a result, the regularization mechanism in Eq. 7.8 can negatively affect the model performance. Hence, we propose to separately model the residual $y_i^t - q_i^t$ with the aim to refine the physics-based regularization.

We need to emphasize in Eq. 7.6 that $\mathbf{s}_i^t$ contains the target label y_i^t, which is not accessible in unlabeled datasets. So an initial guess for y_i^t in unlabeled datasets is required. A straightforward way is to use the output of a basic LSTM as an approximation of y_i^t and treat it as a start value. Meanwhile, in our methods, we run lasso regression individually over different basins and each basin i has its own unique β_i. For ungaged (unlabeled) basins, we build a forward neural network (FNN) with dropout to approximate the unknown β_i. The input of the FNN is the static feature of basins, and the output is the β_i associated with the basin.

Inspired by prior work [96], we also augment the input feature and treat the pseudo label as an extra feature. Hence, the basic goal of the hybrid data model is to combine the physics and neural network input to overcome their complementary deficiencies and leverage information in both physics and data. Meanwhile, if there are systematic discrepancies (biases) in the physics input, then the hybrid data model can learn to complement them by extracting complex features from the space of neural network input and thus reducing

our knowledge gaps. The new input feature vector for LSTM is $\mathbf{x}^t_{aug,i} = (\mathbf{x}^t_i, q^t_i)$. In labelled datasets, we denote

$$\hat{res}^t_i = \beta_{i,0} + \mathbf{s}^t_i \cdot \beta_i \tag{7.10}$$

and the new corrected pseudo label is

$$\hat{q}^t_i = q^t_i - \hat{res}^t_i \tag{7.11}$$

and we can update $\mathbf{x}^t_{aug,i} = (\mathbf{x}^t_i, \hat{q}^t_i)$.

In unlabeled basins, we have approximated β_i obtained from the FNN because we cannot run lasso regression in unlabeled datasets. Meanwhile, $\mathbf{s}^t_i = (\mathbf{x}^t_i, y^t_i)$ and y^t_i is missing, we use $\hat{y}^t_i$ from basic LSTM as an initial guess (Fig. 7.1). Therefore, we get $\mathbf{s}^t_i = (\mathbf{x}^t_i, \hat{y}^t_i)$ in unlabeled datasets, and then we use the same way to calculate $\hat{q}^t_i$ and update $\mathbf{x}^t_{aug,i}$ according to Eq. 7.10 and 7.11.

7.4.4 Training between the Source Domain (Corrected Pseudo Labels) and the Target Domain (Real Labels)

An alternative way of using simulated data in ML training is to pre-train the ML with simulated labels and then fine-tune it with observed labels [91]. The main drawback of pretraining is domain shifting [21] indicating that too much fine-tuning done in the target domain can break the smooth pattern existing in the pre-trained model and thus cause over-fitting. To address this issue, we propose a new method to train the model on pseudo labels and real labels back and forth. We can train the model on the source domain for a few iterations, move to the target domain and then move back.

Inspired by the above approach, we start training from corrected pseudo labels $\hat{q}^t_i$ obtained through Eq. 7.11, and then train the model on real labels. After that, we correct pseudo labels $\hat{q}^t_i$ using new predictions $\hat{y}^t_i$ and repeat the process again and again (Fig. 7.2).

7.5 Experiments

We evaluate the proposed method for predicting streamflow using the continental hydrology data set, CAMELS [1]. The CAMELS data set contains continuous meteorologic input, observed streamflow data, and catchment dependent spatially varying but temporally physical descriptors. CAMELS encompasses a total of 671 watersheds across the contiguous US. In our experiments, we use 480 basins for evaluation. One half of basins are for training and the other half for testing.

Then we describe baselines and discuss the results about the predictive performance using the effec-

Algorithm 6 The flow of the proposed iterative training switch model.

initialize the LSTM network model
for $epoch = 1$: number of iterations **do**
  for $k = 1$: number of training iterations on q **do**
    for $t = 1$: T, $i = 1$: M **do**
      calculate q_i^t through (Eq. 7.7)
      make residual correction through (Eq. 7.10)
      train the model using $\hat{q}_i^t$ through (Eq. 7.11)
  for $l = 1$: number of training iterations on y **do**
    for $t = 1$: T, $i = 1$: N **do**
      train the model using y_i^t
      get $\hat{y}_i^t$ and prepare for (Eq. 7.10)

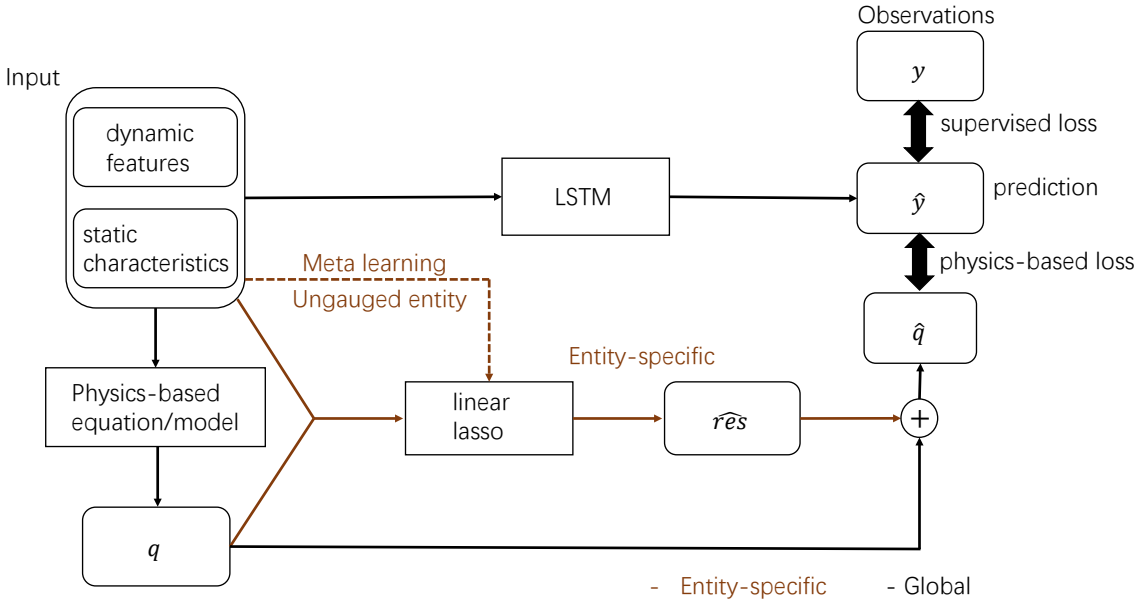


Figure 7.1: Diagram for physics input + residual correction method.

tiveness of pre-training, residual correction and model generalization. All experiments are conducted using TensorFlow on a computer with the following configuration: Intel Core i7-8750H CPU @2.20GHz × 6 Processor, 16 GiB Memory, GeForce GTX 1060, 64-bit Win10 OS.

7.5.1 Baselines

7.5.1.1 Pretrain + Fine-tuning

We pretrain our LSTM model on all pseudo labels and then train it on real labels. The pretrain process takes 50 epochs. In order to find the optimal fine-tuning epochs, we early stop the fine-tuning process and plot the testing error step by step in Table 7.1.

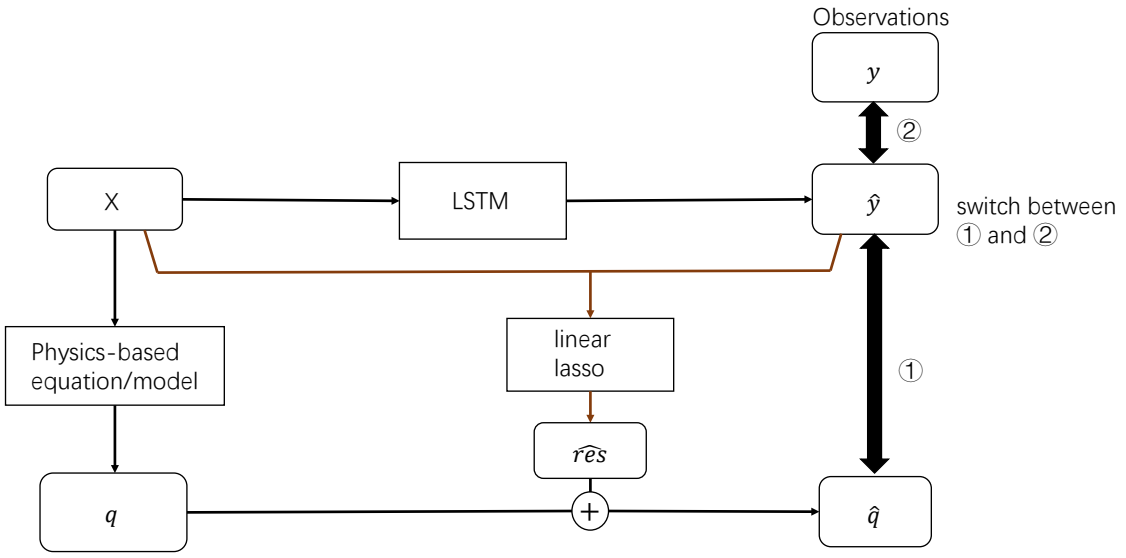


Figure 7.2: Diagram for iteration on source domain and target domain + residual correction.

7.5.1.2 Training between Source Domain (Pseudo Labels) and Target Domain (Real Labels)

We train the model on source domain for a few iterations, move to target domain and then move back. It takes some effort to decide how many epochs of training needed on both domains in order to achieve the best performance.

7.5.1.3 Perturbation to the Proposed Methods

To test the robustness of our approaches, we perturb the evaporation and soil water data in Eq. 7.6 and test the model's performance and robustness.

7.5.1.4 LSTM + Tradaboost

[42] This is a transfer learning baseline with the aim to transfer the knowledge from simulated data to real data. It integrates LSTM with instance-based transfer learning through tradaboost [47].

Tradaboost extends boosting-based learning algorithms and allows users to utilize a small amount of newly labelled data to leverage the old data to construct a high-quality model for the new data.

7.5.2 Results

7.5.2.1 Pretrain with Early Stop Fine-tuning

In table 7.1, we can achieve the best performance of the model using a few fine-tuning epochs. For fine-tuning with fewer data (60 basins), we can adopt the pretrain model directly to avoid overfitting to this small dataset. While, given more data (120, 240 basins), we can let the model learn some real patterns through more

fine-tuning epochs. For 120 basins, 10 epochs fine-tuning is the best. For 240 basins, 30 epochs fine-tuning achieves the best performance.

7.5.2.2 General Performance for All Methods

Table 7.2 summarizes the general performance of all methods mentioned in this chapter. For regularization methods, the performance can be quite similar with switch iteration methods, but far worse with fewer data. It is because the regularization methods do not utilize the whole set of pseudo labels, making the training dataset very small. However, with large datasets, regularization methods can even outperform some switch iteration methods due to the augmented input data. The physics variable and feature variable can overcome their complementary deficiencies and leverage information in both physics and data. The fine-tuning based methods are more stable with fewer data, but it can suffer from domain shifting.

7.5.2.3 Regularization Based Methods

Table 7.3 summarizes the performance of all regularization methods. The basic LSTM performs the worst, except regularization on all pseudo labels. The basic form of regularization can help LSTM stay consistent with the pseudo label so that it improves the prediction accuracy.

Meanwhile, we use regularization on all pseudo label to further handle the data insufficiency problem, and the outcome is straight forward. With 60 basins data, the result is quite similar to pretrain methods and provide a more stable prediction than basic model. On the contrary, as we increase the size of training datasets, the predictions get worse. It is because the effect of regularization is so strong, such that it breaks the true pattern that the model should learn from the real labels.

In order to improve the efficiency of regularization, we add the residual correction to our model. The RMSE reduced in all aspect with various training size. Furthermore, we add the corrected pseudo label to the model input instead of using them in the regularization term solely, and thus further improve our results. Regularization + physics input + residual correction model provides the best results we can obtain at the current stage.

7.5.2.4 Domain Switch Based Methods

Beside the regularization methods, we also utilize the domain switch iteration methods to prevent overfitting. In table 7.4, we compare 3 methods mentioned in this chapter. The plain pretrain + fine-tuning model obviously overfits in small datasets, but achieves an average performance as the basic LSTM model in a full dataset. The iteration methods can prohibit the overfitting by switching training domains and preserve the pattern existing in pseudo label domain. It can achieve better results in all aspects. Finally, our proposed

Table 7.1: RMSE of streamflow prediction using different fine-tuning epochs in pretrain and fine-tuning model. We compare the performance by using real labels from 60, 120 and 240 basins.

basins	0 shot	10 epochs	20 epochs	30 epochs	40 epochs
60	2.316	2.535	2.751	3.234	2.878
120	2.357	2.105	2.206	2.651	2.562
240	2.245	2.013	2.020	1.945	2.137

Table 7.2: RMSE using different numbers of training basins for 50 training epochs. Reg denotes regularization. Phy denotes physics input. Corr and correction denote residual correction.

Method	60 basins	120 basins	240 basins
basic LSTM	11.871	9.384	2.082
LSTM + regularization	10.923	9.036	2.005
pretrain + fine-tuning	4.164	3.037	2.092
LSTM + tradaboost	2.688	2.587	2.302
LSTM + reg + phy + corr	13.791	9.951	1.937
iteration on domains	2.138	2.168	1.933
iteration + correction	2.098	2.133	1.976

Table 7.3: Performance of regularization-based methods using different numbers of training basins

Method	60 basins	120 basins	240 basins
basic LSTM	11.871	9.384	2.082
LSTM + regularization	10.923	9.036	2.005
LSTM + reg (all data)	3.011	3.130	3.236
LSTM + reg + corr	9.527	5.703	1.962
LSTM + reg + phy + corr	13.791	9.951	1.937

iteration + residual correction methods can make an even better prediction in small datasets (60, 120 basins) which are the best performance we can obtain. And a comparable good prediction using full dataset because the corrected pseudo labels do not play a significant role in the training given sufficient real labels.

7.5.2.5 Switch Method Robustness

This table 7.5 summarizes the model robustness and performance under noise perturbation. We add 0.02 standard deviation to evapotranspiration data to test the robustness of the proposed method. The traditional domain switch iteration method does not show strong robustness to perturbation, since there are an obvious RMSE rising up in 60 and 240 basins. However, in our proposed iteration + residual correction method, it shows strong stability handling noisy data in large training datasets.

7.5.2.6 Ablation Study

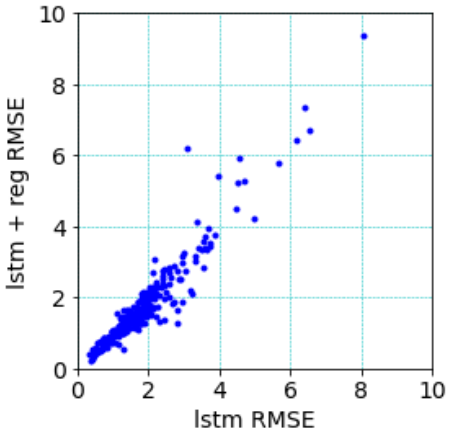
Fig. 7.3(a), 7.3(b), 7.4(a) and 7.4(b) shows the ablation study between different methods. Each dot represents a test basin and the value in x-y axis denotes RMSE.

Table 7.4: Performance of pre-training-based methods using different numbers of training basins

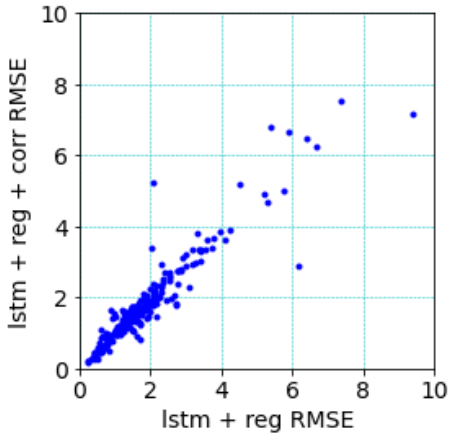
Method	60 basins	120 basins	240 basins
pretrain + fine-tuning	4.164	3.037	2.092
iteration on domains	2.138	2.168	1.933
iteration + correction	2.098	2.133	1.976

Table 7.5: Model robustness to noise using different numbers of training basins

Method	60 basins	120 basins	240 basins
iteration on domains	2.138	2.168	1.933
iteration + noise(0.02 std)	2.361	2.192	2.018
iteration + corr + noise(0.02 std)	2.588	2.144	1.949

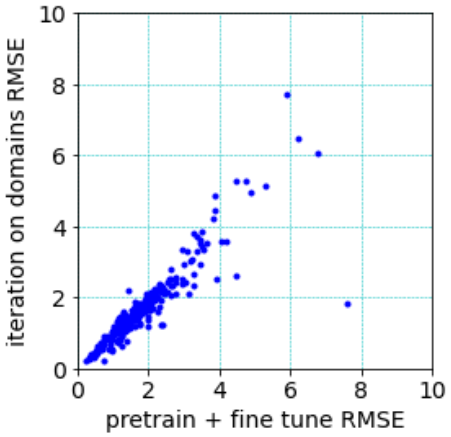


(a) 1. LSTM 2. LSTM + regularization

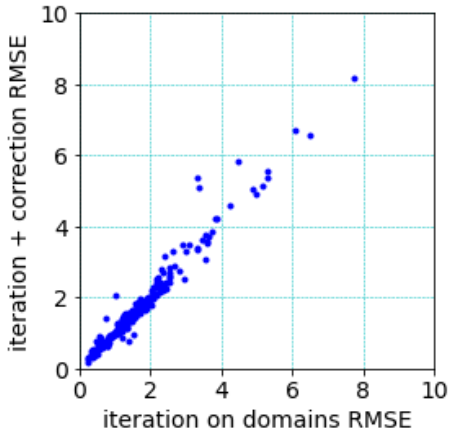


(b) 1. LSTM + regularization 2. LSTM + regularization + residual correction

Figure 7.3:

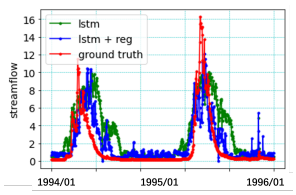


(a) 1. pretrain + fine-tuning 2.iteration on domains

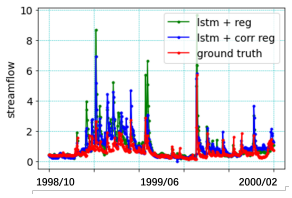


(b) 1. iteration on domains 2. iteration + residual correction

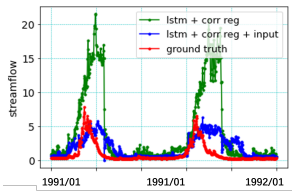
Figure 7.4:



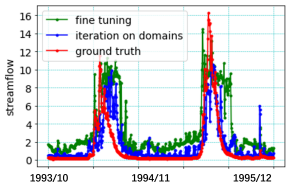
(a) 1. LSTM 2. LSTM + regularization



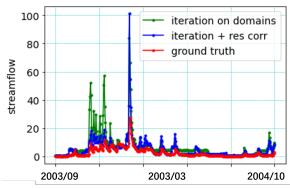
(b) 1. LSTM + regularization 2. LSTM + regularization + residual correction



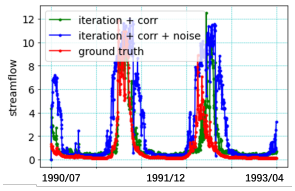
(c) 1. LSTM + regularization + residual correction 2. LSTM + regularization + residual correction + physics input



(d) 1. pretrain + fine tuning 2. iteration on source & target domains



(e) 1. iteration on source and target domains 2. iteration on source & target domains + residual correction



(f) 1. iteration on source and target domains + residual correction 2. iteration on source & target domains + residual correction + noisy data

Figure 7.5: Streamflow predictions using different methods. Figure 7.5(a) is in basin 462. Figure 7.5(b) is in basin 214. Figure 7.5(c) is in basin 472. Figure 7.5(d) is in basin 462. Figure 7.5(e) is in basin 492. Figure 7.5(f) is in basin 462.

7.6 Conclusion

In this chapter, we propose two methods to handle transfer learning under inaccurate physics laws. To deal with the inaccuracy caused by the physics law, we use a lasso regression to correct the pseudo label and let them rejoin the training process. Our methods show strong resistance to data absence and noise interference and a predictive performance comparable to that of the model using actual physical descriptors.

Future results can include how to more precisely approximate the residue, since we are only using a regression method with regularization. We may try transformer or other types of temporal neural network models to further improve our results.

CHAPTER 8

List of Publications

8.1 Published

- **Tianshu Bao**, Shengyu Chen, Taylor T Johnson, Peyman Givi, Shervin Sammak, Xiaowei Jia, "Physics Guided Neural Networks for Spatio-temporal Super-resolution of Turbulent Flows", the 38th Conference on Uncertainty in Artificial Intelligence (UAI), 2022, August.

- **Tianshu Bao**, Xiaowei Jia, Jacob Zwart, Jeffrey Sadler, Alison Appling, Samantha Oliver, Taylor T. Johnson, "Partial Differential Equation Driven Dynamic Graph Networks for Predicting Stream Water Temperature", In 2021 IEEE International Conference on Data Mining (ICDM), pp. 11-20, 2021, December.

- Shengyu Chen, **Tianshu Bao**, Peyman Givi, Can Zheng, Xiaowei Jia, "Reconstructing Turbulent Flows Using Physics-Aware Spatio-Temporal Dynamics and Test-Time Refinement", ACM Transactions on Intelligent Systems and Technology (TIST), accepted, arXiv preprint arXiv:2304.12130

- Hoang-Dung Tran, **Tianshu Bao**, Taylor T. Johnson, "Discrete-Space Analysis of Partial Differential Equations (PDEs) (Benchmark Proposal)", In 5th Applied Verification for Continuous and Hybrid Systems Workshop (ARCH), Oxford, UK, 2018, July

8.2 Submitted

- **Tianshu Bao**, Weiming Xiang and Taylor T. Johnson, "Partial Differential Hybrid Automaton", submitted to Hybrid System Control and Computing

CHAPTER 9

Appendix

9.1 Reachability Analysis for Partial Differential Equations

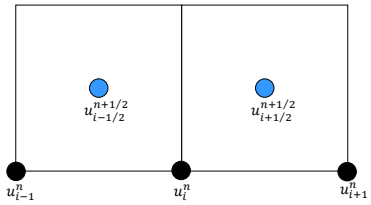


Figure 9.1: Computational stencils of the predictor of the Richtmyer two-step method . Predicted solutions are shown in light blue.

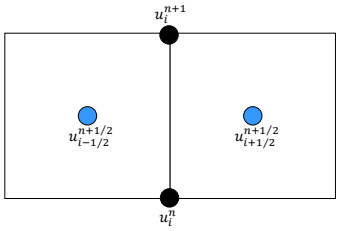


Figure 9.2: Computational stencils of the corrector of the Richtmyer two-step method . Predicted solutions are shown in light blue.

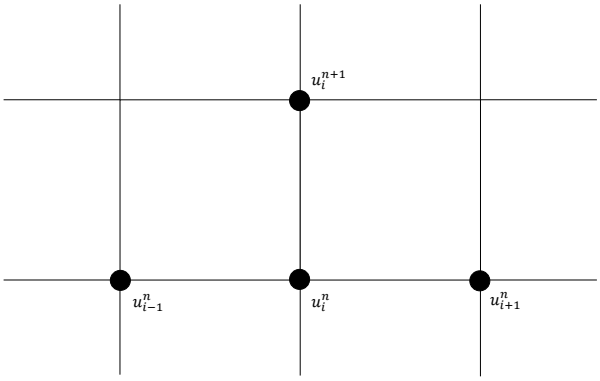


Figure 9.3: In the scalar case of (4.8), the target function is u_i^{n+1} and the variables are u_{i-1}^n, u_i^n and u_{i+1}^n.

The method is stable only if the *Courant-Friedrichs-Lewy* condition (CFL) is satisfied [182] pages 232–238 which has the form below.

$$\left| \frac{\lambda_j \Delta t}{\Delta x} \right| \leq 1. \tag{9.1}$$

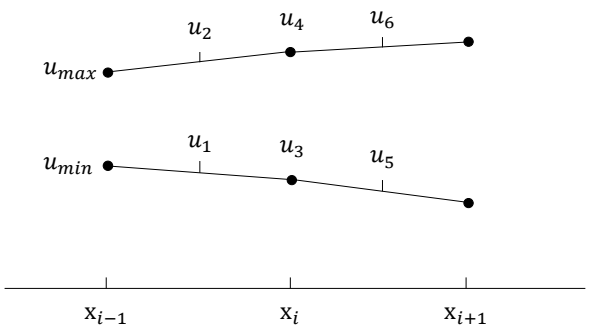


Figure 9.4: The diagram shows the points appearing on u_{max} and u_{min} layer. u_1, u_2, u_5 and u_6 are obtained through interpolation .

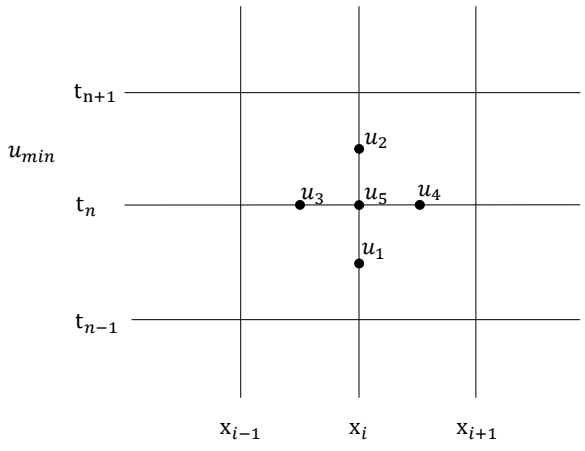


Figure 9.5: Diagram for $u_1, u_2, ...u_5$ on u_{min} lay. u_1, u_2, u_3 and u_4 are obtained through interpolation

The general form of the one-sided method is of the form:

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\Delta x}\lambda_j(v_i^n - v_{i-1}^n),\ \lambda_j > 0, \tag{9.2}$$

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\Delta x}\lambda_j(v_{i+1}^n - v_i^n),\ \lambda_j < 0. \tag{9.3}$$

Theorem 9.1.1. *The iteration procedure of (9.2) and (9.3) is monotone if the CFL condition is satisfied.*

Proof. Assume there are two lists of values:

$u_{max}^n = (u_{max,1}^n, u_{max,2}^n, u_{max,3}^n, ..., u_{max,m}^n)$,

$u_{min}^n = (u_{min,1}^n, u_{min,2}^n, u_{min,3}^n, ..., u_{min,m}^n)$.

For every i ($1 \le i \le m$), $u_{max,i}^n > u_{min,i}^n$. We want to show that $u_{max,i}^{n+1} > u_{min,i}^{n+1}$ for every $i, 1 \le i \le m$, holds as well.

Assuming $\lambda_j > 0$, the following two equations are constructed from (9.2).

$$u_{max,i}^{n+1} = u_{max,i}^n - \frac{\Delta t}{\Delta x}\lambda_i(u_{max,i}^n - u_{max,i-1}^n), \tag{9.4}$$

$$u_{min,i}^{n+1} = u_{min,i}^n - \frac{\Delta t}{\Delta x}\lambda_i(u_{min,i}^n - u_{min,i-1}^n). \tag{9.5}$$

Use (9.4) - (9.5), resulting in (9.6)

$$K_i^{n+1} = K_i^n - r(K_i^n - K_{i-1}^n), \tag{9.6}$$

where K_i^n denotes $(u_{max,i}^n - u_{min,i}^n)$ and r denotes $\frac{\lambda_i \Delta t}{\Delta x}$. The following is the result of reordering (9.6).

$$K_i^{n+1} = (1-r)K_i^n + rK_{i-1}^n, \tag{9.7}$$

where $(1-r) \ge 0$, according to the CFL condition (9.1). Also $r > 0$ since $\lambda_i > 0$. Therefore, $(1-r)K_i^n \ge 0$ and $rK_{i-1}^n > 0$. Concluding that

$$\begin{aligned} K_i^{n+1} &> 0, \\ u_{max,i}^{n+1} &> u_{min,i}^{n+1}. \end{aligned} \tag{9.8}$$

The same proof can be done for $\lambda_j < 0$. □

Theorem 9.1.2. *The iteration procedure of (9.13) is monotone if A is a scalar and CFL condition is satisfied.*

Proof. We create two lists as above. The equation with maximum value list is:

$$\begin{aligned} u_{max,i}^{n+1} &= \frac{1}{2}(u_{max,i+1}^{n} + u_{max,i-1}^{n}) \\ &- \frac{\Delta t}{2\Delta x} A(u_{max,i+1}^{n} - u_{max,i-1}^{n}). \end{aligned} \tag{9.9}$$

Correspondingly, the equation with minimum value list is:

$$\begin{aligned} u_{min,i}^{n+1} &= \frac{1}{2}(u_{min,i+1}^{n} + u_{min,i-1}^{n}) \\ &- \frac{\Delta t}{2\Delta x} A(u_{min,i+1}^{n} - u_{min,i-1}^{n}). \end{aligned} \tag{9.10}$$

Using (9.9) - (9.10), we obtain the equation

$$K_i^{n+1} = \frac{1}{2}(K_{i+1}^{n} + K_{i-1}^{n}) - \frac{\Delta t}{2\Delta x} A(K_{i+1}^{n} - K_{i-1}^{n}). \tag{9.11}$$

Simplifying (9.11) results in

$$\begin{aligned} K_i^{n+1} &= \frac{1}{2}(K_{i+1}^{n} + K_{i-1}^{n}) - \frac{\Delta t}{2\Delta x} A(K_{i+1}^{n} - K_{i-1}^{n}), \\ &= (\frac{1}{2} - \frac{r}{2})K_{i+1}^{n} + (\frac{1}{2} + \frac{r}{2})K_{i-1}^{n}, \\ &= \frac{1-r}{2}K_{i+1}^{n} + \frac{1+r}{2}K_{i-1}^{n}, \end{aligned} \tag{9.12}$$

where $r = \frac{A\Delta t}{\Delta x}$.

The stability requirement for the *Lax-Friedrich* method is also the CFL condition (9.1) [182]. This leads to the conclusion that $K_i^{n+1} > 0$. Therefore, the scalar form of (9.13) is monotone.

□

The *Lax-Friedrichs* scheme [110] is given below.

$$u_i^{n+1} = \frac{1}{2}(u_{i+1}^{n} + u_{i-1}^{n}) - \frac{\Delta t}{2\Delta x} A(u_{i+1}^{n} - u_{i-1}^{n}) \tag{9.13}$$

where u_t, u_x are approximated and u_i^n is an approximate value of $u(x_i, t_n)$.

Theorem 9.1.3. *The iteration procedure of (9.13) is monotone if A is a decoupled system and CFL condition is satisfied.*

Proof.

$$K_i^{n+1} = \frac{1}{2}(K_{i+1}^{n} + K_{i-1}^{n}) - \frac{\Delta t}{2\Delta x} A(K_{i+1}^{n} - K_{i-1}^{n}) \tag{9.14}$$

Matrix A is the $k \times k$ decoupled system and K_i^n is $k \times 1$ vector representing $u_{max} - u_{min}$ at $x = x_i, t = t_n$.

After simplification, the similar form is obtained using maximum and minimum lists.

$$K_i^{n+1} = \frac{I-R}{2}K_{i+1}^n + \frac{I+R}{2}K_{i-1}^n \tag{9.15}$$

where $R = \frac{A\Delta t}{\Delta x}$.

The CFL condition for system of equations is

$$max_{1\leq j\leq n}\left|\lambda_j\right|\frac{\Delta t}{\Delta x} < 1 \tag{9.16}$$

where λ_j is the j^{th} eigenvalue of A.

All eigenvalues of $I+R$ and $\frac{I+R}{2}$ are greater than 0 and these eigenvalues are the diagonal entries of $\frac{I+R}{2}$. Therefore, all values of vector $\frac{I+R}{2}K_{i-1}^n$ are positive.

A similar proof towards $I-R$ can be done in the same way. Thus, all of the elements of K_i^{n+1} are greater than 0 and (9.13) is monotone.

□

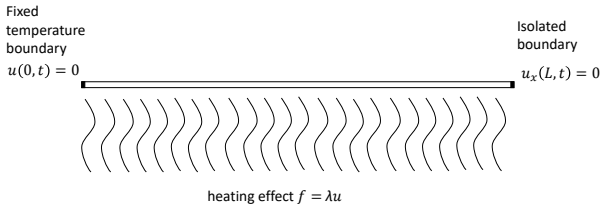


Figure 9.6: One dimension heat equation model.

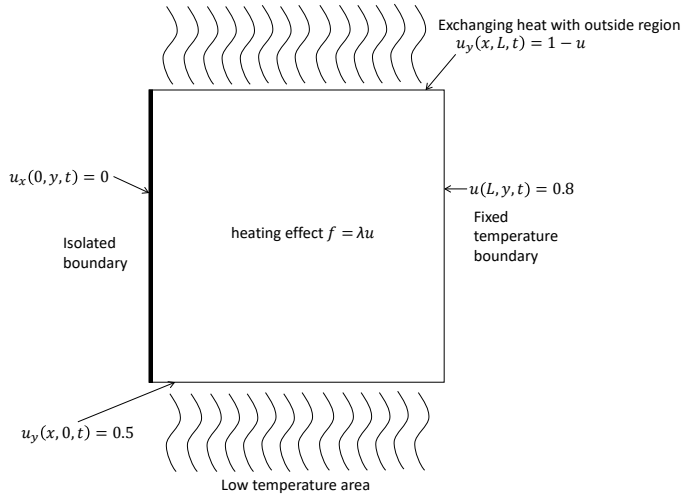


Figure 9.7: Two dimension heat equation model.

Algorithm 7 Arnoldi Iteration

given random nonzero b, let $q_1 = b/\|b\|$
for $k = 1, 2, 3, \ldots$ **do**
    $v = Aq_k$
    for $j = 1$ to k **do**
        $h_{jk} = q_j^T v$
        $v = v - h_{jk} q_j$
    $h_{k+1,k} = \|v\|$
    $q_{k+1} = v/h_{k+1,k}$

References

[1] N. Addor, A. J. Newman, N. Mizukami, and M. P. Clark. The camels data set: catchment attributes and meteorology for large-sample studies. *Hydrology and Earth System Sciences*, 21(10):5293–5313, 2017.

[2] S. Aditya, Y. Yang, and C. Baral. Integrating knowledge and reasoning in image understanding. *arXiv preprint arXiv:1906.09954*, 2019.

[3] N. Ahn, B. Kang, and K.-A. Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network, 2018.

[4] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[5] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[6] L. I. Allerhand and U. Shaked. Robust stability and stabilization of linear switched systems with dwell time. *IEEE Transactions on Automatic Control*, 56(2):381–386, 2010.

[7] M. Althoff. An introduction to cora 2015. In *Proc. of the workshop on applied verification for continuous and hybrid systems*, pages 120–151, 2015.

[8] M. Althoff and B. H. Krogh. Zonotope bundles for the efficient computation of reachable sets. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6814–6821. IEEE, 2011.

[9] M. Althoff and B. H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Transactions on Automatic Control*, 59(2):371–383, 2013.

[10] B. Anderson, T. S. Hy, and R. Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519, 2019.

[11] O. Andersson, F. Heintz, and P. Doherty. Model-based reinforcement learning in continuous environments using real-time constrained optimization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[12] L. Arsenault, A. Lopez-Bezanilla, O. von Lilienfeld, and A. Millis. Machine learning for many-body physics: The case of the anderson impurity model. *Phys. Rev. B*, 2014.

[13] S. Bak and P. S. Duggirala. Simulation-equivalent reachability of large linear systems with inputs. In *International Conference on Computer Aided Verification*, pages 401–420. Springer, 2017.

[14] N. Banagaaya, G. Alı, and W. Schilders. *Index-aware model order reduction methods*. Springer, 2016.

[15] T. Bao, S. Chen, T. T. Johnson, P. Givi, S. Sammak, and X. Jia. physics guided neural networks for spatio-temporal super-resolution of turbulent flows. In *38th Conference on Uncertainty in Artificial Intelligence (UAI)*, Aug. 2022.

[16] T. Bao, S. Chen, T. T. Johnson, P. Givi, S. Sammak, and X. Jia. Physics guided neural networks for spatio-temporal super-resolution of turbulent flows. In *Uncertainty in Artificial Intelligence*, pages 118–128. PMLR, 2022.

[17] T. Bao, X. Jia, J. Zwart, J. Sadler, A. Appling, S. Oliver, and T. T. Johnson. Partial differential equation driven dynamic graph networks for predicting stream water temperature. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 11–20. IEEE, 2021.

[18] C. K. Batchelor and G. Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[19] A. M. Bayen, R. L. Raffard, and C. J. Tomlin. Network congestion alleviation using adjoint hybrid control: Application to highways. In *International Workshop on Hybrid Systems: Computation and Control*, pages 95–110. Springer, 2004.

[20] H. E. Beck, A. I. van Dijk, A. De Roo, D. G. Miralles, T. R. McVicar, J. Schellekens, and L. A. Bruijnzeel. Global-scale regionalization of hydrologic model parameters. *Water Resources Research*, 52(5):3599–3622, 2016.

[21] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

[22] L. E. Besaw, D. M. Rizzo, P. R. Bierman, and W. R. Hackett. Advances in ungauged streamflow prediction using artificial neural networks. *Journal of Hydrology*, 386(1-4):27–37, 2010.

[23] K. Beven and J. Freer. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the glue methodology. *Journal of hydrology*, 249(1-4):11–29, 2001.

[24] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. *BMVA press*, 2012.

[25] G. Blöschl and M. Sivapalan. Scale issues in hydrological modelling: a review. *Hydrological processes*, 9(3-4):251–290, 1995.

[26] G. Blöschl, M. Sivapalan, T. Wagener, A. Viglione, and H. Savenije. *Runoff prediction in ungauged basins: synthesis across processes, places and scales*. Cambridge University Press, 2013.

[27] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *PNAS*, 2007.

[28] G.-J. Both, S. Choudhury, P. Sens, and R. Kusters. Deepmod: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, 428:109985, 2021.

[29] M. E. Brachet, D. Meiron, S. Orszag, B. Nickel, R. Morf, and U. Frisch. The taylor-green vortex and fully developed turbulence. *Journal of Statistical Physics*, 34(5):1049–1063, 1984.

[30] J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model. *Journal of Computational Science*, 44:101171, 2020.

[31] J. R. Brett. Energetic responses of salmon to temperature. a study of some thermal relations in the physiology and freshwater ecology of sockeye salmon (oncorhynchus nerkd). *American zoologist*, 11(1):99–113, 1971.

[32] S. Brunton, J. Proctor, and J. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 2016.

[33] J. Butcher. Runge-kutta methods. *Scholarpedia*, 2(9):3147, 2007.

[34] G. Byrne and P. Ponzi. Differential-algebraic systems, their applications and solutions. *Computers & chemical engineering*, 12(5):377–382, 1988.

[35] H. Cai, V. W. Zheng, and K. C.-C. Chang. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*, 2017.

[36] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[37] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[38] S. Chen, S. Sammak, P. Givi, J. P. Yurko, and X. Jia. Reconstructing high-resolution turbulent flows using physics-guided neural networks. *arXiv preprint arXiv:2109.03327*, 2021.

[39] S. Chen, J. A. Zwart, and X. Jia. Physics-guided graph meta learning for predicting water temperature and streamflow in stream networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2752–2761, 2022.

[40] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[41] Y. Chen, Y. Tai, X. Liu, C. Shen, and J. Yang. Fsrnet: End-to-end learning face super-resolution with facial priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2492–2501, 2018.

[42] Z. Chen, H. Xu, P. Jiang, S. Yu, G. Lin, I. Bychkov, A. Hmelnov, G. Ruzhnikov, N. Zhu, and Z. Liu. A transfer learning-based lstm strategy for imputing large-scale consecutive missing data and its application in a water quality prediction system. *Journal of Hydrology*, 602:126573, 2021.

[43] W. Cheng, M. Zhao, Z. Ye, and S. Gu. Mfagan: A compression framework for memory-efficient on-device super-resolution gan, 2021.

[44] C. G. Claudel and A. M. Bayen. Solutions to switched hamilton-jacobi equations and conservation laws using hybrid components. In *International Workshop on Hybrid Systems: Computation and Control*, pages 101–115. Springer, 2008.

[45] E. A. Cross and I. M. Mitchell. Level set methods for computing reachable sets of systems with differential algebraic equation dynamics. In *2008 American Control Conference*, pages 2260–2265. IEEE, 2008.

[46] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang. Second-order attention network for single image super-resolution. In *CVPR*, 2019.

[47] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning.(2007), 193–200. In *Proceedings of the 24th international conference on Machine learning*, 2007.

[48] T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In *International Conference on Formal Methods in Computer-Aided Design*, pages 21–36. Springer, 2004.

[49] P. A. Davidson. *Turbulence: an introduction for scientists and engineers*. Oxford university press, 2015.

[50] A. Daw, A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[51] Z. Deng, C. He, Y. Liu, and K. C. Kim. Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. *Physics of Fluids*, 31(12):125111, 2019.

[52] M. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

[53] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[54] S. J. Dugdale, D. M. Hannah, and I. A. Malcolm. River temperature modelling: A review of process-based approaches and future directions. *Earth-Science Reviews*, 175:97–113, 2017.

[55] V. V. Duong, T. N. Huu, J. Yim, and B. Jeon. A fast and efficient super-resolution network using hierarchical dense residual learning. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1809–1813, 2021.

[56] S. Džeroski, L. Todorovski, I. Bratko, B. Kompare, and V. Križman. Equation discovery with ecological applications. In *Machine learning methods for ecological applications*, pages 185–207. Springer, 1999.

[57] E. Eich-Soellner and C. Führer. *Numerical methods in multibody dynamics*, volume 45. Springer, 1998.

[58] L. C. Evans. Partial differential equations. *American Mathematical Society*, 2010.

[59] G. Evensen et al. *Data assimilation: the ensemble Kalman filter*, volume 2. Springer, 2009.

[60] K. Fang and C. Shen. Near-real-time forecast of satellite-based soil moisture using long short-term memory with an adaptive data integration kernel. *Journal of Hydrometeorology*, 21(3):399–413, 2020.

[61] R. M. Felder and R. Brent. Active learning: An introduction. *ASQ higher education brief*, 2(4):1–5, 2009.

[62] A. Fettweis. Robust numerical integration using wave-digital concepts. *Multidimensional Systems and Signal Processing*, 17(1):7–25, 2006.

[63] J. Fish and T. Belytschko. *A first course in finite elements*. Wiley, 2007.

[64] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395. Springer, 2011.

[65] K. Fukami, K. Fukagata, and K. Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[66] K. Fukami, K. Fukagata, and K. Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909, Dec 2020.

[67] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[68] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[69] H. Gao, Z. Wang, L. Cai, and S. Ji. Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[70] G. Garner, I. A. Malcolm, J. P. Sadler, and D. M. Hannah. What causes cooling water temperature gradients in a forested stream reach? *Hydrology and Earth System Sciences*, 18(12):5361, 2014.

[71] A. Geiss and J. C. Hardin. Invertible cnn-based super resolution with downsampling awareness. *arXiv preprint arXiv:2011.05586*, 2020.

[72] D. Gerwin. Information processing, data inferences, and scientific generalization. *Behavioral Science*, 19(5):314–325, 1974.

[73] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes*, 41(2):8966–8971, 2008.

[74] P. Givi. Model-free simulations of turbulent reactive flows. *Progress in Energy and Combustion Science*, 15(1):1–107, 1989.

[75] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[76] D. Graham-Rowe, D. Goldston, C. Doctorow, M. Waldrop, C. Lynch, F. Frankel, R. Reid, S. Nelson, D. Howe, S. Rhee, et al. Big data: science in the petabyte era. *Nature*, 455(7209):8–9, 2008.

[77] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[78] H. V. Gupta, T. Wagener, and Y. Liu. Reconciling theory with observations: elements of a diagnostic approach to model evaluation. *Hydrological Processes: An International Journal*, 22(18):3802–3813, 2008.

[79] J. Han, A. Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *PNAS*, 2018.

[80] J. Han, L. Zhang, and E. Weinan. Solving many-electron schrödinger equation using deep neural networks. *Journal of Computational Physics*, 399:108929, 2019.

[81] Z. Han and B. Krogh. Reachability analysis of hybrid control systems using reduced-order models. In *Proceedings of the 2004 American Control Conference*, volume 2, pages 1183–1189. IEEE, 2004.

[82] P. C. Hanson, A. B. Stillman, X. Jia, A. Karpatne, H. A. Dugan, C. C. Carey, J. Stachelek, N. K. Ward, Y. Zhang, J. S. Read, et al. Predicting lake surface water phosphorus dynamics using process-guided machine learning. *Ecological Modelling*, 430:109136, 2020.

[83] F. M. Hante, G. Leugering, and T. I. Seidman. Modeling and analysis of modal switching in networked transport systems. *Applied Mathematics and Optimization*, 59(2):275–292, 2009.

[84] A. Harten, J. M. Hyman, P. D. Lax, and B. Keyfitz. On finite-difference approximations and entropy conditions for shocks. *Communications on pure and applied mathematics*, 29(3):297–322, 1976.

[85] B. Heinrich. *Finite difference methods on irregular networks: a generalized approach to second order elliptic problems*. Springer, 1987.

[86] J. P. Hespanha and A. S. Morse. Stability of switched systems with average dwell-time. In *Proceedings of the 38th IEEE conference on decision and control (Cat. No. 99CH36304)*, volume 3, pages 2655–2660. IEEE, 1999.

[87] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[88] M. Hrachowitz, H. Savenije, G. Blöschl, J. McDonnell, M. Sivapalan, J. Pomeroy, B. Arheimer, T. Blume, M. Clark, U. Ehret, et al. A decade of predictions in ungauged basins (pub)—a review. *Hydrological sciences journal*, 58(6):1198–1255, 2013.

[89] K.-l. Hsu, H. V. Gupta, and S. Sorooshian. Artificial neural network modeling of the rainfall-runoff process. *Water resources research*, 31(10):2517–2530, 1995.

[90] X. Jia, B. Lin, J. Zwart, J. Sadler, A. Appling, S. Oliver, and J. Read. Graph-based reinforcement learning for active learning in real time: An application in modeling river networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 621–629. SIAM, 2021.

[91] 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 SIAM International Conference on Data Mining*, 2019.

[92] X. Jia, J. Zwart, J. Sadler, A. Appling, S. Oliver, S. Markstrom, J. Willard, S. Xu, M. Steinbach, J. Read, and V. Kumar. Physics-guided recurrent graph model for predicting flow and temperature in river networks. In *SIAM International Conference on Data Mining*. SIAM, 2021.

[93] M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. In *1997 European Control Conference (ECC)*, pages 2005–2010. IEEE, 1997.

[94] K. D. Julian, M. J. Kochenderfer, and M. P. Owen. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics*, 42(3):598–608, 2019.

[95] A. Karpatne, G. Atluri, J. 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 TKDE*, 2017.

[96] A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[97] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[98] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

[99] S. K. Kauwe, J. Graser, A. Vazquez, and T. D. Sparks. Machine learning prediction of heat capacity for solid inorganics. *Integrating Materials and Manufacturing Innovation*, 7(2):43–51, 2018.

[100] Y. Khoo, J. Lu, and L. Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6(1):1, 2019.

[101] D.-Y. Kim, J. Park, and A. M. Morrison. A model of traveller acceptance of mobile technology. *International Journal of Tourism Research*, 10(5):393–407, 2008.

[102] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger. Rainfall–runoff modelling using long short-term memory (lstm) networks. *Hydrology and Earth System Sciences*, 22(11):6005–6022, 2018.

[103] F. Kratzert, D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter, and G. Nearing. Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrology and Earth System Sciences*, 23(12):5089–5110, 2019.

[104] S. N. Krishna and A. Trivedi. Hybrid automata for formal modeling and verification of cyber-physical systems. *arXiv preprint arXiv:1503.04928*, 2015.

[105] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[106] I. Lagaris, A. Likas, and D. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw. Learn*, 1998.

[107] J. H. Lagergren, J. T. Nardini, G. Michael Lavigne, E. M. Rutter, and K. B. Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A*, 476(2234):20190800, 2020.

[108] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[109] P. Langley. Data-driven discovery of physical laws. *Cognitive Science*, 5(1):31–54, 1981.

[110] P. D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on pure and applied mathematics*, 7(1):159–193, 1954.

[111] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[112] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[113] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[114] E. A. Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[115] D. Li and H. Ji. Syntax-aware multi-task graph convolutional networks for biomedical relation extraction. In *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*, pages 28–33, 2019.

[116] X. Li and Y. Guo. Adaptive active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2013.

[117] X. Li, A. Khandelwal, X. Jia, K. Cutler, R. Ghosh, A. Renganathan, S. Xu, K. Tayal, J. Nieber, C. Duffy, et al. Regionalization in a global hydrologic deep learning model: from physical descriptors to random vectors. *Water Resources Research*, 58(8):e2021WR031794, 2022.

[118] M. J. Lighthill and G. B. Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proc. R. Soc. Lond. A*, 229(1178):317–345, 1955.

[119] J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech*, 2016.

[120] B. Liu, J. Tang, H. Huang, and X.-Y. Lu. Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of Fluids*, 2020.

[121] N. Ma, S. Mazumder, H. Wang, and B. Liu. Entity-aware dependency-based deep graph attention network for comparative preference classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5782–5788, 2020.

[122] J. J. Magnuson, L. B. Crowder, and P. A. Medvick. Temperature as an ecological resource. *American Zoologist*, 19(1):331–343, 1979.

[123] M. Margaliot and G. Langholz. Necessary and sufficient conditions for absolute stability: the case of second-order systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 50(2):227–234, 2003.

[124] S. L. Markstrom, R. S. Regan, L. E. Hay, R. J. Viger, R. M. Webb, R. A. Payn, and J. H. LaFontaine. Prms-iv, the precipitation-runoff modeling system, version 4. *US Geological Survey Techniques and Methods*, 2015.

[125] R. März. Canonical projectors for linear differential algebraic equations. *Computers & Mathematics with Applications*, 31(4-5):121–135, 1996.

[126] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, 34(6):85–95, 2017.

[127] J. J. McDonnell and K. Beven. Debates—the future of hydrological sciences: A (common) path forward? a call to action aimed at understanding velocities, celerities and residence time distributions of the headwater hydrograph. *Water Resources Research*, 2014.

[128] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[129] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic control*, 50(7):947–957, 2005.

[130] A. S. Morse. Supervisory control of families of linear set-point controllers-part i. exact matching. *IEEE transactions on Automatic Control*, 41(10):1413–1431, 1996.

[131] Z. Moshe, A. Metzger, G. Elidan, F. Kratzert, S. Nevo, and R. El-Yaniv. Hydronets: Leveraging river structure for hydrologic modeling. *arXiv preprint arXiv:2007.00595*, 2020.

[132] L. Muñoz, X. Sun, R. Horowitz, and L. Alvarez. Traffic density estimation with the cell transmission model. In *American Control Conference, 2003. Proceedings of the 2003*, volume 5, pages 3750–3755. IEEE, 2003.

[133] N. Muralidhar, J. Bu, Z. Cao, L. He, N. Ramakrishnan, D. Tafti, and A. Karpatne. Phynet: Physics guided neural networks for particle drag force prediction in assembly. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 559–567. SIAM, 2020.

[134] N. Muralidhar, M. Islam, M. Marwah, A. Karpatne, and N. Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *IEEE Big Data*. IEEE, 2018.

[135] S. L. Neitsch, J. G. Arnold, J. R. Kiniry, and J. R. Williams. Soil and water assessment tool theoretical documentation version 2009. Technical report, Texas Water Resources Institute, 2011.

[136] F. Noé, A. Tkatchenko, K.-R. Müller, and C. Clementi. Machine learning for molecular simulation. *Annual review of physical chemistry*, 71:361–390, 2020.

[137] O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowlishwaran. Surfnet: Super-resolution of turbulent flows with transfer learning using small datasets. *arXiv preprint arXiv:2108.07667*, 2021.

[138] S. A. Orszag and G. S. Patterson. Numerical simulation of turbulence. In *Statistical Models and Turbulence*, pages 127–147, New York, NY, 1986. Springer-Verlag.

[139] H. Owhadi. Bayesian numerical homogenization. *Multiscale Modeling & Simulation*, 13(3):812–828, 2015.

[140] H. Owhadi, C. Scovel, and T. Sullivan. Brittleness of bayesian inference under finite information in a continuous world. *Electronic Journal of Statistics*, 9(1):1–79, 2015.

[141] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine*, pages 21–36, 2003.

[142] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

[143] S. Pettersson and B. Lennartson. Stabilization of hybrid systems using a min-projection strategy. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, volume 1, pages 223–228. IEEE, 2001.

[144] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[145] C. Prieto, N. Le Vine, D. Kavetski, E. García, and R. Medina. Flow prediction in ungauged catchments using probabilistic random forests regionalization and new statistical adequacy tests. *Water Resources Research*, 55(5):4364–4392, 2019.

[146] Y. Qi, Q. Li, H. Karimian, and D. Liu. A hybrid model for spatiotemporal forecasting of pm2. 5 based on graph convolutional neural network and long short-term memory. *Science of the Total Environment*, 2019.

[147] R. Rai and C. K. Sahu. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. *IEEE Access*, 8:71050–71073, 2020.

[148] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 335:736–746, 2017.

[149] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683–693, 2017.

[150] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[151] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Numerical gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, 2018.

[152] M. Raissi, A. Yazdani, and G. Karniadakis. Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data. *arXiv:1808.04327*, 2018.

[153] J. Rauch and M. Reed. Jump discontinuities of semilinear, strictly hyperbolic systems in two variables: Creation and propagation. *Communications in Mathematical Physics*, 81(2):203–227, 1981.

[154] T. Razavi and P. Coulibaly. Streamflow prediction in ungauged basins: review of regionalization methods. *Journal of hydrologic engineering*, 18(8):958–975, 2013.

[155] E. K. Read, L. Carr, L. De Cicco, H. A. Dugan, P. C. Hanson, J. A. Hart, J. Kreft, J. S. Read, and L. A. Winslow. Water quality data for national-scale aquatic research: The water quality portal. *Water Resources Research*, 53(2):1735–1745, 2017.

[156] J. S. Read, X. Jia, J. Willard, A. P. Appling, J. A. Zwart, S. K. Oliver, A. Karpatne, G. J. Hansen, P. C. Hanson, W. Watkins, et al. Process-guided deep learning predictions of lake water temperature. *Water Resources Research*, 55(11):9173–9190, 2019.

[157] R. S. Regan, S. L. Markstrom, L. E. Hay, R. J. Viger, P. A. Norton, J. M. Driscoll, and J. H. LaFontaine. Description of the national hydrologic model for use with the precipitation-runoff modeling system (prms). Technical report, US Geological Survey, 2018.

[158] S. H. Rudy, J. N. Kutz, and S. L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

[159] K. Ryan, J. Lengyel, and M. Shatruk. Crystal structure prediction via deep learning. *Journal of the American Chemical Society*, 140(32):10158–10168, 2018.

[160] I. S. Strub and A. M. Bayen. Weak formulation of boundary conditions for scalar conservation laws: An application to highway traffic modelling. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 16(16):733–748, 2006.

[161] P. Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2006.

[162] S. Sahoo, C. Lampert, and G. Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

[163] L. Samaniego, R. Kumar, and S. Attinger. Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. *Water Resources Research*, 46(5), 2010.

[164] M. J. Sanders, S. L. Markstrom, R. S. Regan, and R. D. Atkinson. Documentation of a daily mean stream temperature module—an enhancement to the precipitation-runoff modeling system. Technical report, US Geological Survey, 2017.

[165] Science. Special online collection: dealing with data. *Science*, 331(6018), 2011.

[166] J. Seibert. Regionalisation of parameters for a conceptual rainfall-runoff model. *Agricultural and forest meteorology*, 98:279–293, 1999.

[167] T. J. Sejnowski, P. S. Churchland, and J. A. Movshon. Putting big data to good use in neuroscience. *Nature neuroscience*, 17(11):1440, 2014.

[168] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[169] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.

[170] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[171] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.

[172] G. Singh, T. Gehr, M. Püschel, and M. Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

[173] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[174] M. Sivapalan, K. Takeuchi, S. Franks, V. Gupta, H. Karambiri, V. Lakshmi, X. Liang, J. McDonnell, E. Mendiondo, P. O'connell, et al. Iahs decade on predictions in ungauged basins (pub), 2003–2012: Shaping an exciting future for the hydrological sciences. *Hydrological sciences journal*, 48(6):857–880, 2003.

[175] G. D. Smith, G. D. Smith, and G. D. S. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

[176] K. Stengel, A. Glaws, D. Hettinger, and R. N. King. Adversarial super-resolution of climatological wind and solar data. *Proceedings of the National Academy of Sciences*, 117(29):16805–16815, 2020.

[177] W. A. Strauss. *Partial differential equations: An introduction*. Wiley, 2007.

[178] U. G. Survey. National water information system data available on the world wide web (usgs water data for the nation). *http://waterdata.usgs.gov/nwis/*, 2016.

[179] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2798, 2017.

[180] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[181] F. Theurer, K. Voos, and W. Miller. Instream water temperature model. instream flow information paper 16. us fish wildl serv. *Div. Biol. Serv., Tech. Rep. FWS OBS*, 84(15):11–42, 1984.

[182] J. W. Thomas. *Numerical partial differential equations: finite difference methods*, volume 22. Springer Science & Business Media, 2013.

[183] C. J. Tomlin, J. Lygeros, and S. S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.

[184] H.-D. Tran, F. Cai, M. L. Diego, P. Musau, T. T. Johnson, and X. Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

[185] H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson. Star-based reachability analysis of deep neural networks. In *International symposium on formal methods*, pages 670–686. Springer, 2019.

[186] H.-D. Tran, P. Musau, D. M. Lopez, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson. Parallelizable reachability analysis algorithms for feed-forward neural networks. In *2019 IEEE/ACM 7th International Conference on Formal Methods in Software Engineering (FormaliSE)*, pages 51–60. IEEE, 2019.

[187] H.-D. Tran, L. V. Nguyen, N. Hamilton, W. Xiang, and T. T. Johnson. Reachability analysis for high-index linear differential algebraic equations. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 160–177. Springer, 2019.

[188] H.-D. Tran, L. V. Nguyen, W. Xiang, and T. T. Johnson. Order-reduction abstractions for safety verification of high-dimensional linear systems. *Discrete Event Dynamic Systems*, 27(2):443–461, 2017.

[189] H.-D. Tran, W. Xiang, T. T. Johnson, and S. Bak. Reachability analysis for one dimensional linear parabolic equations. *6th IFAC Conference on Analysis and Design of Hybrid Systems*, 2018.

[190] *Reynolds Stress Field and Turbulent Kinetic Energy Budget in a Repeating Compressor Stage*, volume Volume 2E: Turbomachinery of *Turbo Expo: Power for Land, Sea, and Air*, 09 2020. V02ET41A017.

[191] U. Upadhyay and S. P. Awate. Robust super-resolution gan, with manifold-based and perception loss. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 1372–1376, 2019.

[192] T. Venkatesh, R. Srivastava, P. Bhatt, P. Tyagi, and R. K. Singh. A comparative study of various deep learning techniques for spatio-temporal super-resolution reconstruction of forced isotropic turbulent flows. *arXiv preprint arXiv:2107.03361*, 2021.

[193] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394*, 2019.

[194] J.-X. Wang, J.-L. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3):034603, 2017.

[195] X. Wang, K. Yu, C. Dong, and C. C. Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform, 2018.

[196] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCV Workshops*, 2018.

[197] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[198] H. Wei, S. Zhao, Q. Rong, and H. Bao. Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods. *International Journal of Heat and Mass Transfer*, 127:908–916, 2018.

[199] J. F. Wendt. *Computational Fluid Dynamics*. Springer, Berlin, Heidelberg, 3rd edition, 1995.

[200] Z. Wenlong, L. Yihao, C. Dong, and Y. Qiao. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. *TPMAI*, 2021.

[201] Y. Wu, Y. Xu, A. Singh, Y. Yang, and A. Dubrawski. Active learning for graph neural networks via node feature propagation. *arXiv preprint arXiv:1910.07567*, 2019.

[202] W. Xiang. On equivalence of two stability criteria for continuous-time switched systems with dwell time constraint. *Automatica*, 54:36–40, 2015.

[203] W. Xiang. Necessary and sufficient condition for stability of switched uncertain linear systems under dwell-time constraint. *IEEE Transactions on Automatic Control*, 61(11):3619–3624, 2016.

[204] W. Xiang. Parameter-memorized lyapunov functions for discrete-time systems with time-varying parametric uncertainties. *Automatica*, 87:450–454, 2018.

[205] W. Xiang, J. Lam, and J. Shen. Stability analysis and l1-gain characterization for switched positive systems under dwell-time constraint. *Automatica*, 85:1–8, 2017.

[206] W. Xiang, H.-D. Tran, and T. T. Johnson. On reachable set estimation for discrete-time switched linear systems under arbitrary switching. In *2017 American Control Conference (ACC)*, pages 4534–4539. IEEE, 2017.

[207] W. Xiang, H.-D. Tran, and T. T. Johnson. Output reachable set estimation for switched linear systems and its application in safety verification. *IEEE Transactions on Automatic Control*, 62(10):5380–5387, 2017.

[208] W. Xiang, H.-D. Tran, and T. T. Johnson. Robust exponential stability and disturbance attenuation for discrete-time switched systems under arbitrary switching. *IEEE Transactions on Automatic Control*, 63(5):1450–1456, 2017.

[209] T. Xie and J. C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[210] Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

[211] Z. Xu, H. Su, P. Shi, R. Lu, and Z.-G. Wu. Reachable set estimation for markovian jump neural networks with time-varying delays. *IEEE transactions on cybernetics*, 47(10):3208–3217, 2016.

[212] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[213] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[214] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS computational biology*, 16(11):e1007575, 2020.

[215] Y. Ye, S. Hou, L. Chen, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao. Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection. In *IJCAI*, pages 4150–4156, 2019.

[216] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[217] L. Zhang and H. Gao. Asynchronously switched control of switched linear systems with average dwell time. *Automatica*, 46(5):953–958, 2010.

[218] L. Zhang and W. Xiang. Mode-identifying time estimation and switching-delay tolerant control for switched systems: An elementary time unit approach. *Automatica*, 64:174–181, 2016.

[219] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018.

[220] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution, 2018.

[221] S. Zhou, Q. Chen, and X. Wang. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 120:536–546, 2013.

[222] D. Zhu et al. Understanding place characteristics in geographic contexts through graph convolutional neural networks. *Annals of the American Association of Geographers*, 2020.