UNIVERSITY OF CALIFORNIA SAN DIEGO

Thermodynamically Consistent Physics-Informed Data-Driven Computing and Reduced-Order Modeling of Nonlinear Materials

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Structural Engineering with a Specialization in Computational Science

by

Xiaolong He

Committee in charge:

Professor Jiun-Shyan Chen, Chair
Professor Randolph E. Bank
Professor Boris Kramer
Professor Kenneth Loh
Professor Shabnam J. Semnani

2022

University of California San Diego

2022

DEDICATION

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, Prof. J. S. Chen, for his constant support, advice, and encouragement throughout my Ph.D. studies. It has been my great honor and experience to learn from Prof. Chen. I am always inspired to become a better researcher by his devotion and attitude to research.

I would like to acknowledge my defense and candidacy committee members, Prof. Randolph E. Bank, Prof. Boris Kramer, Prof. Kenneth Loh, and Prof. Shabnam J. Semnani for their insightful comments and valuable suggestions to my research. I would like to express my gratitude to Prof. Yuri Bazilevs, Prof. Petr Krysl, Prof. Robert Asaro, Prof. Chung-Hao Lee, Prof. John Hodgson, Prof. Usha Sinha, and Prof. Shantanu Sinha for their research guidance and support.

I would like to thank Dr. Youngsoo Choi, Dr. Jon Belof, Dr. Charles Jekel, Kevin Huynh, William Fries, Nicolas Castrillon, Dr. Siu Wun Cheung, Dr. Dylan Copeland, Jessie Lauzon, Dr. Rebecca Fushimi, Dr. Yixiao Wang, Dr. Ross Kunz, and Dr. Yu-Yen Chen for their great support and guidance during my research internships.

I would also like to deliver many thanks to my great colleagues in the computational mechanics group: Prof. Qizhi He, Dr. Haoyan Wei, Dr. Jonghyuk Baek, Prof. Tsung-Hui Huang, Dr. Jacob Koester, Dr. Marco Pasetto, Karan Taneja, Ryan Schlinkman, Kristen Susuki, Prof. Shigeki Kaneko, Dr. Frank Beckwith, Prof. Michael Hillman, Dr. Yantao Zhang, and many others.

Thanks also go to my nice friends and colleagues: Dr. Chao-Hsien Li, Dr. Heng Liu, Prof. Zhijian Qiu, Prof. Zongliang Du, Prof. Yewei Zheng, Dr. Wenyong Rong, Dr. Jianyu Cheng, Dr. Long Wang, Dr. Lei Li, Dr. Ranting Cui, Dr. Lan Luo, Dr. Rui Kou, Yening Shu, Dr. Kuan-Po Lin, Dr. Yu Lu, Dr. Raghavendra Sivapuram, Dr. Poorya Mirkhosravi, Dr. Giwon Hong, Dr. Marco Pigazzini, Prof. George Moutsanidis, Ricardo Bustamante Hernández, Dr. Rodrigo Chavez Morales, Dr. Mukesh Ramancha, Paulina Díaz-Montiel, Dr. Mohammed Alamer, Douglas de Aquino Castro, Lin Sun, Wenpei Ma, Xinlun Zhao, and many others.

Last but not least, I would like to thank my families for their endless love and support. Without them, I could not have made it this far.

The support by National Science Foundation, National Institutes of Health, Lawrence Livermore National Laboratory, and Idaho National Laboratory is greatly appreciated.

Chapter 3, in part, is a reprint of the material as it appears in: “Xiaolong He and Jiun-Shyan Chen. Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, page 115348, 2022”. The dissertation author is the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the materials as they appear in: “Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, and Shantanu Sinha. Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids. *Data-Centric Engineering*, 1, 2020” and ”Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021”. The dissertation author is the primary investigator and author of these papers.

Chapter 5, in part, is a reprint of the material as it appears in: “Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, and Shantanu Sinha. Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids. *Data-Centric Engineering*, 1, 2020”. The dissertation author is the primary investigator and author of this paper.

Chapter 6, in part, is a reprint of the material as it appears in: “Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021”. The dissertation author is the primary investigator and author of this paper.

Chapter 7, in part, is a reprint of the material as it appears in: “Xiaolong He and Jiun-Shyan Chen. Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. *Computer Methods in Applied Mechanics* *and Engineering*, page 115348, 2022”. The dissertation author is the primary investigator and author of this paper.

Chapter 8, in part, has been submitted for publication of the material as it appears in: “Xiaolong He, Youngsoo Choi, William D. Fries, Jon Belof, and Jiun-Shyan Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 2022”. The dissertation author is the primary investigator and author of this paper.

VITA

2022	Ph.D. in Structural Engineering with a Specialization in Computational Science, University of California, San Diego
2017–2022	Research Assistant, University of California, San Diego
2017	M.S. in Computational Mechanics, Swansea University and University of Stuttgart
2014–2015	Research Assistant, The University of Hong Kong
2014	B.S. in Theoretical and Applied Mechanics, Sun Yat-Sen University

PUBLICATIONS

- Xiaolong He, Youngsoo Choi, William D. Fries, Jon Belof, and Jiun-Shyan Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 2022. Under Review.

- Karan Taneja, Xiaolong He, Qizhi He, Zhao Xinlun, Lin Yun-An, Loh Kenneth, and Jiun-Shyan Chen. A feature-encoded physics-informed parameter identification neural network for musculo-skeletal systems. *Journal of Biomechanical Engineering*, 2022. Accepted.

- William Fries, Xiaolong He, and Youngsoo Choi. Lasdi: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 2022. Accepted.

- Xiaolong He and Jiun-Shyan Chen. Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, page 115348, 2022

- Yu-Yen Chen, M Ross Kunz, Xiaolong He, and Rebecca Fushimi. Recent progress toward catalyst properties, performance, and prediction with data-driven methods. *Current Opinion in Chemical Engineering*, 37:100843, 2022

- M Ross Kunz, Adam Yonge, Xiaolong He, Rakesh Batchu, Zongtang Fang, Yixiao Wang, Gregory S Yablonsky, Andrew J Medford, and Rebecca R Fushimi. Internal calibration of transient kinetic data via machine learning. *Catalysis Today*, 2022

- Xiaolong He, Karan Taneja, Jiun-Shyan Chen, Chung-Hao Lee, John Hodgson, Vadim Malis, Usha Sinha, and Shantanu Sinha. Multiscale modeling of passive material influences on deformation and force output of skeletal muscles. *International Journal for Numerical Methods in Biomedical Engineering*, page e3571, 2022

- Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021

- Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, and Shantanu Sinha. Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids. *Data-Centric Engineering*, 1, 2020

- Benjamin Reedlunn, Georgios Moutsanidis, Jonghyuk Baek, Tsung-Hui Huang, Jacob Koester, Xiaolong He, Haoyan Wei, Karan Taneja, Yuri Bazilevs, and Jiun-Shyan Chen. Initial simulations of empty room collapse and reconsolidation at the waste isolation pilot plant. In *54th US Rock Mechanics/Geomechanics Symposium*. OnePetro, 2020

- Yantao Zhang, Jiun-Shyan Chen, Qizhi He, Xiaolong He, Ramya R Basava, John Hodgson, Usha Sinha, and Shantanu Sinha. Microstructural analysis of skeletal muscle force generation during aging. *International journal for numerical methods in biomedical engineering*, 36(1):e3295, 2020

- Lingling Hu, Danyang Cai, Gangpeng Wu, Xiaolong He, and Tongxi Yu. Influence of internal pressure on the out-of-plane dynamic behavior of circular-celled honeycombs. *International Journal of Impact Engineering*, 104:64–74, 2017

- Changyong Jiang, Xiaolong He, and Lixi Huang. Equivalent acoustic parameters in a periodical waveguide structure. In *Fluid-Structure-Sound Interactions and Control*, pages 83–88. Springer, 2016

- LingLling Hu, Xiaolong He, Gangpeng Wu, and Tongxi Yu. Dynamic crushing of the circular-celled honeycombs under out-of-plane impact. *International Journal of Impact Engineering*, 75:150–161, 2015

ABSTRACT OF THE DISSERTATION

Thermodynamically Consistent Physics-Informed Data-Driven Computing and Reduced-Order Modeling of Nonlinear Materials

by

Xiaolong He

Doctor of Philosophy in Structural Engineering with a Specialization in Computational Science

University of California San Diego, 2022

Professor Jiun-Shyan Chen, Chair

Physical simulations have influenced the advancements in engineering, technology, and science more rapidly than ever before. However, it remains challenging for effective and efficient modeling of complex linear and nonlinear material systems based on phenomenological approaches which require predefined functional forms. The goal of this dissertation is to enhance the predictivity and efficiency of physical simulations by developing thermodynamically consistent data-driven computing and reduced-order materials modeling methods based on emerging machine learning techniques for manifold learning, dimensionality reduction, sequence learning, and system identification.

For *reversible* mechanical systems, we first develop a new data-driven material solver built upon local convexity-preserving reconstruction to capture anisotropic material behaviors and enable data-driven modeling of nonlinear anisotropic elastic solids. A material anisotropic state characterizing the underlying material orientation is introduced for the manifold learning projection in the material solver. To counteract the curse of dimensionality and enhance the generalization ability of data-driven computing, we employ deep autoencoders to discover the underlying low-dimensional manifold of material database and integrate a convexity-preserving interpolation scheme into the novel autoencoder-based data-driven solver to further enhance efficiency and robustness of data searching and reconstruction during online data-driven computation. The proposed approach is shown to achieve enhanced efficiency and generalization ability over a few commonly used data-driven schemes.

For *irreversible* mechanical systems, we develop a thermodynamically consistent machine learned data-driven constitutive modeling approach for path-dependent materials based on measurable material states, where the internal state variables essential to the material path-dependency are inferred automatically from the hidden state of recurrent neural networks. The proposed method is shown to successfully model soil behaviors under cyclic shear loading using experimental stress-strain data.

Lastly, we develop a non-intrusive accurate and efficient reduced-order model based on physics-informed adaptive greedy latent space dynamics identification (gLaSDI) for general high-dimensional nonlinear dynamical systems. An autoencoder and dynamics identification models are trained simultaneously to discover intrinsic latent space and learn expressive governing equations of simple latent-space dynamics. To maximize and accelerate the exploration of the parameter space for optimal model performance, an adaptive greedy sampling algorithm integrated with a physics-informed residual-based error indicator and random-subset evaluation is introduced to search for optimal training samples on the fly, which outperforms the conventional predefined uniform sampling. Compared with the high-fidelity models of various nonlinear dynamical problems, gLaSDI achieves 66 to 4,417$\times$ speed-up with 1 to 5% relative errors.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Physics-Constrained Data-Driven Computing and Nonlinear Materials Modeling

As the fundamental laws of physics and natural sciences, the thermodynamics laws underlie natural processes of *reversible* or *irreversible* mechanical systems. The thermodynamics first law describes energy conservation of mechanical systems. The thermodynamics second law deals with the directionality of thermodynamic processes. Despite the success and advancements of physical simulations in various scientific fields, it remains challenging for accurate and efficient modeling of nonlinear materials subjected to complex *reversible* or *irreversible* mechanical processes.

Constitutive modeling of nonlinear materials subjected to *reversible* or *irreversible* processes is traditionally based on constitutive or material laws to describe the explicit relationship among strains, stresses, and state variables based on experimental observations, mechanistic hypothesis, and mathematical simplifications. However, the phenomenological modeling process inevitably introduces errors due to limited data and mathematical assumptions in model parameter calibration. Moreover, constitutive laws rely on pre-defined functions and often lack generality to capture full aspects of material behaviors, such as anisotropy, nonlinearity, path- or rate-dependency, etc. For example, it remains challenging to formulate conventional phenomenological constitutive laws for musculoskeletal systems consisting of multiple highly nonlinear and anisotropic biological materials [14, 18–20].

Conventional constitutive modeling of *irreversible* path-dependent material systems typically applies models with evolving internal state variables (ISVs) in addition to the state space of deformation [21, 22]. The ISV constitutive modeling framework has been effectively applied to model various nonlinear solid material behaviors, e.g., elasto-plasticity [23, 24], visco-plasticity [25], and material damage [26]. However, ISVs are often non-measurable, which makes it challenging to define a complete and appropriate set of ISVs for highly nonlinear and complicated materials, e.g., geomechanical materials. Further, the traditional ISV constitutive modeling approach often results in excessive complexities with high computational cost, which is undesirable in practical applications.

With advancements in computing power and significant progresses in data mining, machine learning based data-driven approaches, e.g., deep neural networks (DNNs), have been extensively applied to various fields owing to their strong ability of feature/pattern extraction [27], including constitutive material modeling. However, pure black-box data-driven models mapping inputs to outputs without considering the underlying physics suffer from unstable and inaccurate generalization performance. Further, to model *irreversible* path-dependent materials, the DNN-based constitutive models require fully descriptive material's internal states, which is difficult for materials with highly nonlinear and complicated path-dependent behaviors and limits their applications in practice.

In recent years, a physics-constrained data-driven computational paradigm has been proposed to bypass the constitutive modeling step by formulating the boundary value problems as a new optimization problem to search for a solution that is closest to the material data set subjected to equilibrium and compatibility constraints [2, 28]. Despite promising progress in various application domains, these data-driven computing frameworks cannot effectively handle anisotropic material systems with various anisotropic orientations, i.e., orientations of material anisotropy, due to the fact that the data-driven solvers of these frameworks do not consider appropriate material frame of reference. Moreover, these data-driven approaches could encounter difficulties in applications with high dimensional data when material data sampling involves multidimensional and history-dependent state variables.

1.1.2 Physics-Informed Data-Driven Reduced-Order Modeling

Physical simulations have played an increasingly important role in the advancements of engineering, science, and technology. Many physical processes are mathematically modeled by time-dependent nonlinear partial differential equations (PDEs). As it is difficult or even impossible to obtain analytical solutions for many highly complicated problems, various numerical methods have been developed to approximate the analytical solutions. However, high-fidelity forward physical simulations can be computationally intractable even with high performance computing, prohibiting their applications to problems that require a large number of forward simulations [29–33, 33, 34].

To achieve accurate and efficient physical simulations, various physics-constrained data-driven model reduction techniques have been developed, such as the projection-based reduced-order models (ROM), in which the state fields of the full-order model (FOM) are projected on to a linear or nonlinear subspace to significantly reduce the dimension of the state fields. Popular *linear* projection techniques include the proper orthogonal decomposition (POD) [35], the reduced basis method [36], and the balanced truncation method [37]. Despite the success of the classical linear-subspace ROM in many applications, it is limited to the assumption that intrinsic solution space falls into a low-dimensional subspace, indicating that the solution space has a small Kolmogorov *n*-width. In other words, they cannot be effectively applied to advection-dominated problems with sharp gradients, moving shock fronts, and turbulence.

Most classical physics-constrained data-driven projection-based ROMs aforementioned are *intrusive*, which require plugging the reduced-order solution representation into the discretized system of governing equations. Although the intrusive brings many benefits, such as extrapolation robustness, requirement of less training data, and high accuracy, the implementation of the intrusive ROMs requires not only sufficient understanding of the numerical solver of the high-fidelity simulation, but also access to the source code of the numerical solver.

1.2 Objectives

The objective of this work is to enhance the capability and efficiency of physical simulations by developing *thermodynamically consistent* data-driven computing, materials modeling, and reduced-order modeling methods based on emerging machine learning techniques for manifold learning, dimensionality reduction, sequence learning, and system identification. The major developments in this research are summarized as follows.

- For *reversible* mechanical systems:
  - Development of a new data-driven material solver built upon the local convexity-preserving reconstruction scheme [38] in order to model anisotropic nonlinear elastic solids. To this end, a rotated material database is constructed offline and a two-level data search is integrated into the material solver to capture directional dependent material behaviors during online data-driven computing. The proposed data-driven computing framework is verified by modeling the deflection of a cantilever beam with layers containing different fiber directions and inflation of a cylinder where fiber directions vary along the circumferential direction of the cylinder. The data-driven solutions are compared with the constitutive model-based reference solutions to examine the effectiveness and robustness of the proposed methods.
  - Development of a novel data-driven computing approach to overcome the curse of dimensionality and the lack of generalization in the classical model-free data-driven computing approaches. To this end, we propose to introduce deep autoencoders to achieve two major objectives: dimensionality reduction and generalization of physically meaningful constitutive manifold. In this approach, the autoencoders are first trained in an offline stage to extract a representative low-dimensional manifold

(embedding) of the given material data. Autoencoders also provide effective noise filtering through the data compression processes. The trained autoencoders are then incorporated in the data-driven solver during the online computation with customized local convexity-preserving reconstruction to ensure numerical stability and representative constitutive manifold. Hence, all operations related to distance measure, including the search of the closest material points in the dataset are performed in the learned embedding space, circumventing the difficulties resulting from high dimensionality and data noise. We present two different solvers to perform locally convex reconstruction, and demonstrate the one directly providing interpolation approximation without using decoders outperforms the one that fully uses the encoder-decoder network structure. Furthermore, it is shown that the proposed method is computationally tractable, since the additional autoencoder training is conducted offline and the online data-driven computation mainly involve lower-dimensional variables in the embedding space. The proposed method is applied to biological tissue modeling to demonstrate the enhanced effectiveness and generalization capability of Auto-embedding Data-Driven (AEDD) over the a few commonly used data-driven schemes.

- For *irreversible* mechanical systems:

    Development of a thermodynamically consistent machine-learned internal state variable (ISV) approach for data-driven modeling of path-dependent materials, which relies purely on measurable material states. The first thermodynamics principle is integrated into the model architecture whereas the second thermodynamics principle is enforced by a constraint on the network parameters. In the proposed model, a recurrent neural network (RNN) is trained to infer intrinsic ISVs from its hidden (or memory) state that captures essential history-dependent features of data through a sequential input. The RNN describing the evolution of the data-driven machine-learned ISVs follows the thermodynamics second law. In addition, a deep neural network (DNN) is trained simultaneously to predict the material energy potential given strain, ISVs, and temperature (for non-isothermal processes). Further, model robustness and accuracy is enhanced by introducing *stochasticity* to inputs for model training to account for uncertainties of input conditions in testing. The effectiveness and generalization capability of the proposed method are examined by modeling an elasto-plastic material and undrained soil under cyclic shear loading. A parametric study is conducted to investigate the effects of the number of RNN steps, the internal state dimension, the model complexity, and the strain increment on the model performance.

- For general nonlinear dynamical systems:

  Development of a physics-informed adaptive greedy latent space dynamics identification (gLaSDI) framework for non-intrusive accurate and efficient data-driven reduced-order modeling. To maximize and accelerate the exploration of the parameter space for optimal performance, an adaptive greedy sampling algorithm integrated with a physics-informed residual-based error indicator and random-subset evaluation is introduced to search for the optimal and minimal training samples on-the-fly. The proposed gLaSDI framework contains an autoencoder for nonlinear projection to discover intrinsic latent representations and a set of local dynamics identification (DI) models to capture local latent-space dynamics, which is further exploited by an efficient k-nearest neighbor (KNN) convex interpolation scheme. The autoencoder training and dynamics identification in the gLaSDI take place interactively to achieve an optimal identification of simple latent-space dynamics. The effectiveness and capability of the proposed gLaSDI framework are examined by modeling various nonlinear dynamical problems, including Burgers equations, nonlinear heat conduction, and radial advection. The effects of various factors on model performance are investigated, including the number of nearest neighbors for convex interpolation, the latent-space dimension, the complexity of the DI models, and the size of the parameter space. A performance comparison between uniform sampling and the physics-informed greedy sampling is also presented.

1.3 Outline

The remainder of this dissertation is organized as follows. In Chapter 2, we present an overview on machine learning, data-driven materials modeling, physics-constrained data-driven computing, and data-driven reduced-order modeling. In Chapter 3, we review the basics of thermodynamics. In Chapter 4, we introduce the mathematics and fundamentals of physics-constrained data-driven computing frameworks, including the classical distance-minimizing data-driven (DMDD) computing [2] and the recently developed local convexity data-driven (LCDD) computing [38]. We then introduce the developments for *reversible* mechanical systems. In Chapter 5, we propose a new data-driven material solver built upon the local convexity-preserving reconstruction scheme [38] for modeling anisotropic nonlinear elastic solids. In Chapter 6, we propose a novel auto-embedding data-driven (AEDD) computing approach to overcome the issues related to the curse of dimensionality and the lack of generalization in classical model-free data-driven computing approaches. Deep autoencoders are employed for dimensionality reduction and enhanced generalization capability of physically meaningful constitutive manifold. The development for *irreversible* mechanical systems is introduced in Chapter 7, where a thermodynamically consistent machine-learned ISV approach is developed for data-driven modeling of path-dependent materials, which relies purely on measurable material states. The thermodynamics principles are integrated into the model architecture and training. Intrinsic ISVs are automatically inferred by a recurrent neural network (RNN) from its hidden state that captures essential history-dependent features of data through a sequential input. In Chapter 8, an adaptive greedy latent space dynamics identification (gLaSDI) framework is proposed for accurate, efficient, and robust physics-informed data-driven reduced-order modeling of general nonlinear dynamical systems. An adaptive greedy sampling algorithm integrated with a physics-informed residual-based error indicator and random-subset evaluation is introduced to search for the optimal and minimal training samples on the fly. Lastly, conclusions and discussions for future research are given in Chapter 9.

Chapter 2

Literature Review

In this chapter, we review important concepts and methods related to machine learning, data-driven materials modeling, physics-constrained data-driven computing, and data-driven reduced-order modeling.

2.1 Machine Learning: Feature Learning and Dimensionality Reduction

Artificial intelligence (AI) has demonstrated successful applications in all science and engineering domains, such as developing intelligent software or robotics to automate routine labor, understanding speech or images, making diagnoses in medicine and supporting basic scientific research [27]. AI systems acquire knowledge by extracting features and patterns from raw data, known as machine learning (ML). Fig. 2.1 shows an overview of ML algorithms and their applications. In this section, We briefly review the basic concept of *supervised learning* and *unsupervised learning* and introduce the ML algorithms used in the research of this dissertation.

2.1.1 Supervised Learning

Supervised learning aims to map inputs to target outputs. There are two main tasks of supervised learning, i.e., *classification* where the target outputs are qualitative, e.g., categories or classes, and *regression* where the target outputs are quantitative.

Let us consider an input space, $\mathscr{X} \subset \mathbb{R}^{d_{in}}$, and an underlying function $\mathbf{f}: \mathscr{X} \to \mathscr{Y}$ that

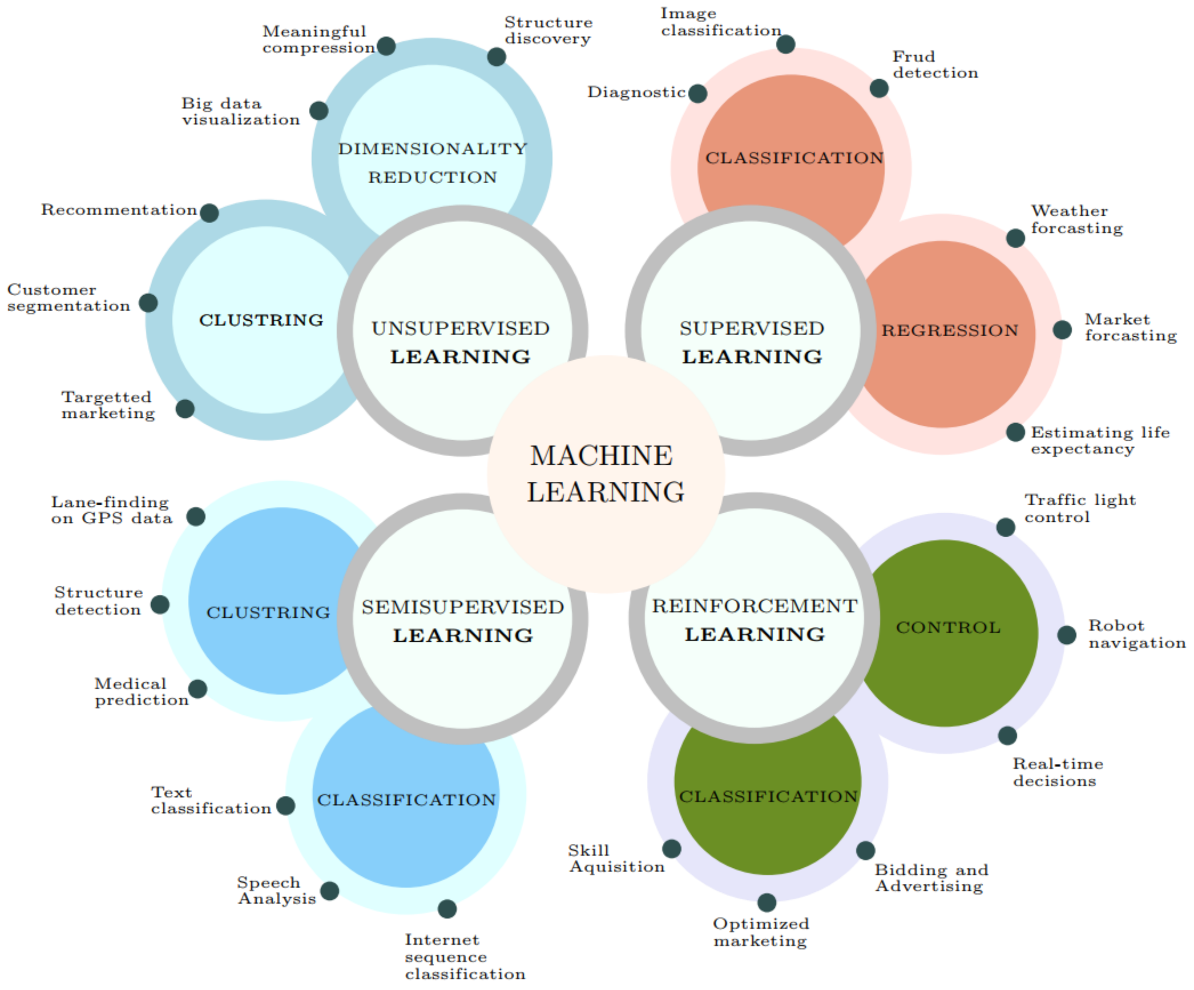


Figure 2.1. An overview of Machine Learning and its applications [1].

maps an input element $\mathbf{x} \in \mathscr{X}$ to an element $\mathbf{y}$ in the output space, $\mathscr{Y} \subset \mathbb{R}^{d_{out}}$. Given input data points sampled from the input space, $\{\mathbf{x}_1,...,\mathbf{x}_M\} \in \mathscr{X}$, and measured outputs, $\{\mathbf{y}_1,...,\mathbf{y}_M\} \in \mathscr{Y}$, supervising learning algorithms aim to find a function $\hat{\mathbf{f}}(\theta) : \mathscr{X} \to \mathscr{Y}$ to approximate the unknown function $\mathbf{f}$, which maps an element $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$ in the input space to an element $\mathbf{y}_i \in \mathbb{R}^{d_{out}}$ in the output space, where $i = 1,...,M$ and θ denotes parameters of $\hat{\mathbf{f}}$, by minimizing a loss (or error) function, $\mathscr{L} : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}_+$, where $\mathbb{R}_+$ denotes a set of non-negative real numbers. For regression tasks, a squared error is often used: $\mathscr{L}(\hat{\mathbf{y}},\mathbf{y}) = ||\hat{\mathbf{y}} - \mathbf{y}||^2_{L_2}$, where $\hat{\mathbf{y}} = \hat{\mathbf{f}}(\mathbf{x};\theta)$. The optimal parameters θ^* of $\hat{\mathbf{f}}$ can be obtained by the following minimization

$$\theta^* = \arg\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} \mathscr{L}(\hat{\mathbf{f}}(\mathbf{x}_i;\theta),\mathbf{y}_i). \tag{2.1}$$

Deep Neural Networks (DNNs)

Deep neural networks (DNNs), as the core of the deep learning [27], represent complex models that relate data inputs, $\mathbf{x} \in \mathbb{R}^{d_{in}}$, to data outputs, $\mathbf{y} \in \mathbb{R}^{d_{out}}$. A typical DNN is composed of an input layer, an output layer, and L hidden layers. Each hidden layer transforms the outputs of the previous layer through an affine mapping followed by a nonlinear activation function $a(\cdot)$, which can be written as:

$$\mathbf{x}^{(l)} = a(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1,...,L, \tag{2.2}$$

where the superscript (l) denotes the layer the quantities belong to, e.g., $\mathbf{x}^{(l)} \in \mathbb{R}^{n_l}$ is the outputs of layer l with n_l neurons. $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ are the weight matrix for linear mapping and the bias vector of layer l, respectively, where $n_0 = d_{in}$ is the input dimension. They are trainable parameters to be optimized through training. For a fully-connected layer, the number of trainable parameters is $(n_{l-1}+1)n_l$.

Commonly used activation functions include the logistic sigmoid, the hyperbolic tangent function, the rectified linear unit (ReLU), and the leaky ReLU [39]. Note that the choice of the activation of the output layer depends on the type of ML tasks. For regression tasks, which is the application of this study, a linear function is used in the output layer where the last hidden layer information is mapped to the output vector $\hat{\mathbf{y}}$, expressed as: $\hat{\mathbf{y}} = \mathbf{W}^{(L+1)}\mathbf{x}^{(L)} + \mathbf{b}^{(L+1)}$, where $\hat{\mathbf{y}}$ denotes the DNN approximation of the data output $\mathbf{y}$. Fig. 2.2 shows the computational graph of a feed-forward DNN with three input neurons, two hidden layers, and two output neurons.

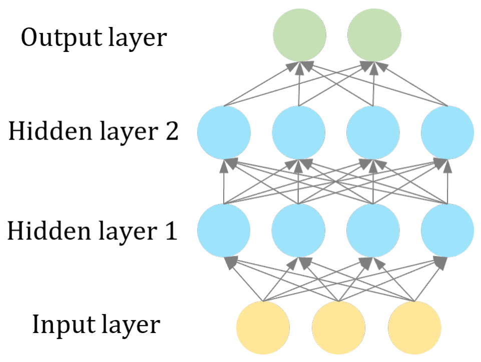


Figure 2.2. Computational graph of a feed-froward deep neural network (DNN) with three input neurons, two hidden layers, and two output neurons.

We denote $\mathbf{f}_{DNN}(\theta)$ as a DNN mapping function, where θ is the collection of all trainable weight and bias coefficients, $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L+1}$. The forward prediction of the DNN can be expressed $\hat{\mathbf{y}} = \mathbf{f}_{DNN}(\mathbf{x};\theta)$. The optimal parameters θ^* of the DNN can be obtained by minimizing the following loss function:

$$\theta^* = \arg\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} ||\mathbf{f}_{DNN}(\mathbf{x}_i;\theta) - \hat{\mathbf{y}}_i||_{L_2}^2 + \beta \sum_{l=1}^{L+1} ||\mathbf{W}^{(l)}||_F^2, \tag{2.3}$$

where β is a regularization parameter, and $||\cdot||_F$ denotes the Frobenius norm. The L_2-norm based weight regularization term is used to prevent over-fitting issues [27, 40].

Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) designed for sequence learning have demonstrated successful applications in various domains, such as machine translation and speech recognition, due to their capability of learning history-dependent features that are essential for sequential prediction [41, 42]. Fig. 2.3 illustrates the computational graphs of a folded RNN and an unfolded RNN, where $\mathbf{h}$ is a hidden state that captures essential history-dependent features from past information, which makes RNNs particularly suitable for modeling path-dependent material behaviors. Unfolding of the RNN computational graph results in *parameter sharing* across the network structure, reducing the number of trainable parameters and thus leading to more efficient training. The length of input/output sequences can be arbitrary, which allows generalization to sequence lengths not appeared in the training set. Each step can be viewed as a state. Despite the history sequence length, the trained RNN model always has the same input size, since it is specified in terms of transition from one state to another rather than in terms of a variable-length history of states [27]. The forward propagation of RNN begins with an initial hidden state and the propagation equations at time step (state) n are defined as

$$\mathbf{h}_n = a_{tanh}\big(\mathbf{W}_{hh}\mathbf{h}_{n-1} + \mathbf{W}_{xh}\mathbf{x}_n + \mathbf{b}_h\big), \tag{2.4a}$$

$$\hat{\mathbf{y}}_n = \mathbf{W}_{hy}\mathbf{h}_n + \mathbf{b}_y, \tag{2.4b}$$

where a_{tanh} is the hyperbolic tangent function; $\mathbf{W}_{xh}, \mathbf{W}_{hh}$, and $\mathbf{W}_{hy}$ are trainable weight coefficients for input-to-hidden, hidden-to-hidden, and hidden-to-output transformations, respectively; $\mathbf{b}_h$ and $\mathbf{b}_y$ are trainable bias coefficients. These trainable parameters are shared across all RNN steps. Eqs. (2.4a) transforms the previous hidden state $\mathbf{h}_{n-1}$ and the current input $\mathbf{x}_n$ to the current hidden state $\mathbf{h}_n$, while (2.4b) transforms the current hidden state $\mathbf{h}_n$ to the current output $\hat{\mathbf{y}}_n$. The history information is captured by the hidden state of RNN by repeating the transformation in Eq. (2.4a) for all RNN steps. The hidden state that carries the essential history-dependent information is passed to the final step and informs the final prediction.

Depending on applications, RNNs can have flexible architectures of input and output, such as one-to-one, one-to-many, many-to-one, and many-to-many [43]. For example, the unfolded RNN shown in Fig. 7.2(a) is a many-to-many type of RNN, which can be applied to, for example, name entity recognition. We denote $\mathbf{f}_{RNN}(\theta)$ as a RNN mapping function, where

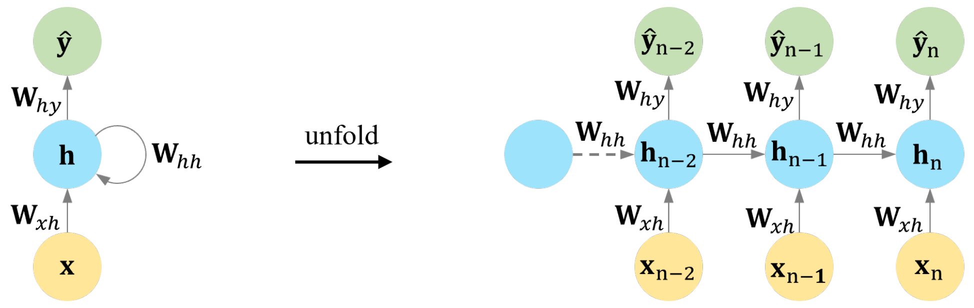


Figure 2.3. Computational graphs of a folded recurrent neural network (RNN) and an unfolded RNN.

θ denotes the trainable parameters. The optimal parameters θ^* of RNN can be obtained by minimizing the following loss function:

$$\theta^* = \arg\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} ||\mathbf{f}_{RNN}(\hat{\mathbf{x}}_i;\theta) - \hat{\mathbf{y}}_i||_{L_2}^2. \tag{2.5}$$

2.1.2 Unsupervised Learning

Unsupervised learning aims to extract hidden patterns or structures in data. The main unsupervised learning tasks include clustering and dimensionality reduction. As the dimension of data increases, the amount of training data required for satisfactory model performance increases exponentially, known as the *curse of dimensionality*. We will introduce autoencoders to counteract the curse of dimensionality in the following.

Autoencoders

Autoencoders [44, 45] are an unsupervised learning technique in which special architectures of DNNs are leveraged for dimensionality reduction or representation learning. Specially, an autoencoder aims to optimally copy its input to output with the most representative features by introducing a low-dimensional embedding layer (or called **a code**). As shown in Fig. 2.4, an autoencoder consists of two parts, an encoder function $\mathbf{h}_{\text{enc}}(\cdot;\theta_{\text{enc}}) : \mathbb{R}^d \to \mathbb{R}^p$ and a decoder function $\mathbf{h}_{\text{dec}}(\cdot;\theta_{\text{dec}}) : \mathbb{R}^p \to \mathbb{R}^d$, such that the autoencoder is

$$\tilde{\mathbf{x}} = \mathbf{h}(\mathbf{x};\theta_{\text{enc}},\theta_{\text{dec}}) := (\mathbf{h}_{\text{dec}} \circ \mathbf{h}_{\text{enc}})(\mathbf{x}) \tag{2.6a}$$

$$:= \mathbf{h}_{\text{dec}}(\mathbf{h}_{\text{enc}}(\mathbf{x};\theta_{\text{enc}});\theta_{\text{dec}}), \tag{2.6b}$$

where $p < d$ is the embedding dimension, θ_{enc} and θ_{dec} are the DNN coefficients of encoder and deconder parts, respectively, and $\tilde{\mathbf{x}}$ is the output of the autoencoder, a reconstruction of the original input $\mathbf{x}$. With the latent dimension p much less than the input dimension d, the encoder $\mathbf{h}_{\text{enc}}$ is trained to learn the compressed representation of $\mathbf{x}$, denoted as the embedding $\mathbf{x}' \in \mathbb{R}^p$, whereas the decoder $\mathbf{h}_{\text{dec}}$ reconstructs the input data by mapping the embedding representation back to the high-dimensional space.

It is important to note that similar to any other dimensionality reduction techniques [46], the employment of autoencoders is based on the *manifold hypothesis*, which presumes that the given high-dimensional input data, e.g., the material dataset $\mathbb{E}$, lies on a low-dimensional manifold $\mathscr{E}'$ that is embedded in a higher-dimensional vector space, as shown by the schematic figures at the bottom of Fig. 2.4.

Given the autoencoder architecture $\mathbf{h}(\cdot;\theta_{\text{enc}},\theta_{\text{dec}})$ in Eq. (2.6), the optimal parameters θ^*_{enc} and θ^*_{dec} can be obtained by minimizing the following loss function:

$$(\theta^*_{\text{enc}},\theta^*_{\text{dec}}) = \underset{\theta_{\text{enc}},\theta_{\text{dec}}}{\arg\min} \frac{1}{M} \sum_{i=1}^{M} ||\mathbf{h}(\mathbf{x}_i;\theta_{\text{enc}},\theta_{\text{dec}}) - \mathbf{x}_i||^2_{L_2} + \beta \sum_{l=1}^{L+1} ||\mathbf{W}^{(l)}||^2_F, \tag{2.7}$$

where $\{\mathbf{W}^{(l)}\}_{l=1}^{L+1}$ are trainable weight coefficients; β is a regularization parameter, and $||\cdot||_F$ denotes the Frobenius norm. Here, the loss function consists of the reconstruction error over all training data and a L_2-norm based weight regularization term used to prevent over-fitting issues [27, 40].

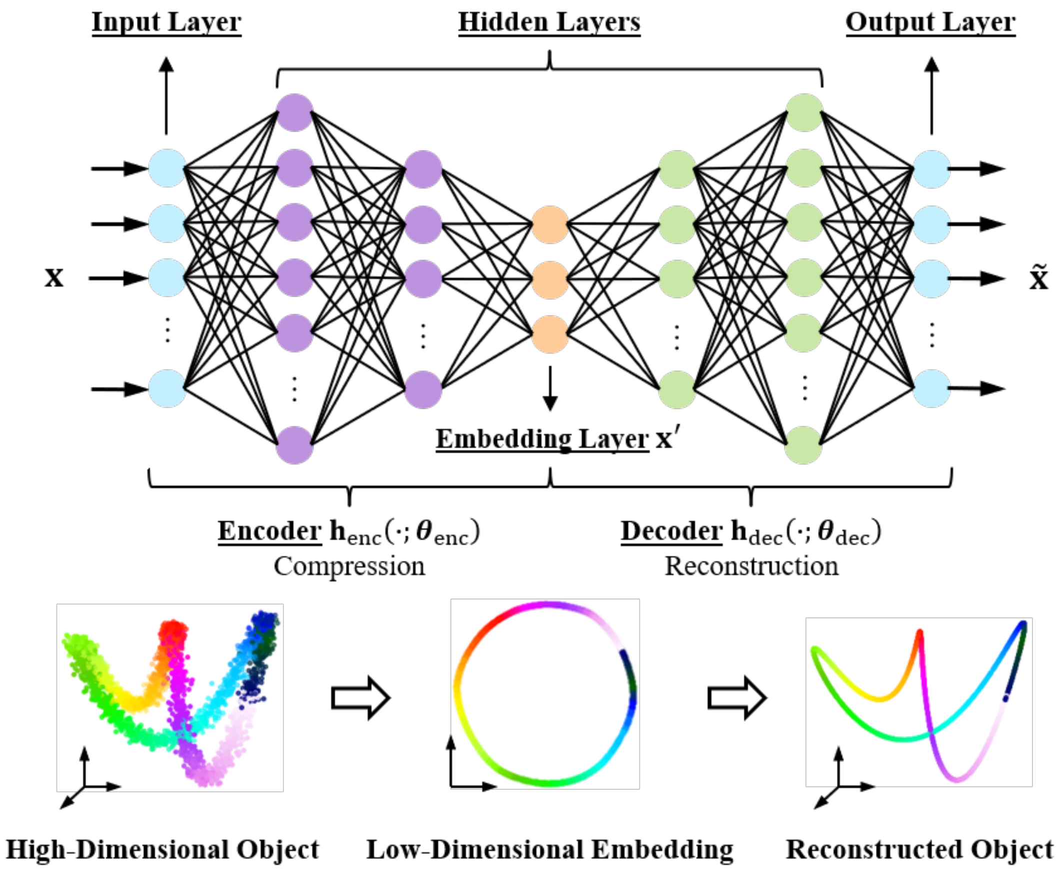

Input Layer
Hidden Layers
Output Layer
$\mathbf{x}$
$\tilde{\mathbf{x}}$
Embedding Layer $\mathbf{x}'$
Encoder $\mathbf{h}_{\text{enc}}(\cdot;\boldsymbol{\theta}_{\text{enc}})$
Compression
Decoder $\mathbf{h}_{\text{dec}}(\cdot;\boldsymbol{\theta}_{\text{dec}})$
Reconstruction
High-Dimensional Object
Low-Dimensional Embedding
Reconstructed Object

Figure 2.4. Schematic of an autoencoder consisting of an encoder and a decoder, where the dimension of the embedding layer is smaller than the input dimension. For a high-dimensional input object, the encoder learns a compressed low-dimensional embedding, on which the decoder optimally reconstructs the input object.

2.2 Data-Driven Materials Modeling

Traditional constitutive modeling is based on constitutive or material laws to describe the explicit relationship among the measurable material states, e.g., stresses and strains, and internal state variables (ISVs) based on experimental observations, mechanistic hypothesis, and mathematical simplifications. However, limited data and functional form assumptions inevitably introduce errors to the model parameter calibration and model prediction. Moreover, with the pre-defined functions, constitutive laws often lack generality to capture full aspects of material behaviors [6, 47].

Path-dependent constitutive modeling typically applies models with evolving ISVs in addition to the state space of deformation [21, 22]. The ISV constitutive modeling framework has been effectively applied to model various nonlinear solid material behaviors, e.g., elasto-plasticity [23, 24], visco-plasticity [25], and material damage [26]. However, ISVs are often non-measurable, which makes it challenging to define a complete and appropriate set of ISVs for highly nonlinear and complicated materials, e.g., geomechanical materials. Further, the traditional ISV constitutive modeling approach often results in excessive complexities with high computational cost, which is undesirable in practical applications.

With advancements in computing power and significant progresses in data mining [27], machine learning (ML) based data-driven approaches have demonstrated successful applications in various engineering problems, such as solving partial differential equations [48–51], system or parameter identification [48, 51–56], reduced-order modeling [7, 9, 57–61], material design [62, 63], structural design [64, 65], damage and fracture modeling [66], etc.

ML models, such as deep neural networks (DNNs), have emerged as a promising alternative for constitutive modeling due to their strong flexibility and capability in extracting complex features and patterns from data [67]. DNNs have been applied to model a variety of materials, including concrete materials [68], hyper-elastic materials [69], visco-plastic material of steel [70], and homogenized properties of composite structures [71]. DNN-based constitutive models haven been integrated into finite element solvers to predict path- or rate-dependent materials behaviors [72–76]. Recently, physical constraints or principles have been integrated into DNNs for data-driven constitutive modeling, including symmetric positive definiteness [77], material frame invariance [78], and thermodynamics [79, 80]. However, to model path-dependent materials, the DNN-based constitutive models require fully understood and prescribed material's internal states, which is difficult for materials with highly nonlinear and complicated path-dependent behaviors and limits their applications in practice.

Recurrent neural networks (RNNs) designed for sequence learning have been successfully applied in various domains, such as machine translation and speech recognition, due to their capability of learning history-dependent features that are essential for sequential prediction [41, 42]. The RNN and gated variants, e.g., the long short-term memory (LSTM) [81] cells and the gated recurrent units (GRUs) [82, 83], have been applied to path-dependent materials modeling [84], including plastic composites [85], visco-elasticity [86], and homogeneous anisotropic hardening [87]. RNN-based constitutive models have also been applied to accelerate multi-scale simulations with path-dependent characteristics [88–92]. Recently, Bonatti and Mohr [93] proposed a self-consistent RNN for path-dependent materials such that the model predictions converge as the loading increment is decreased. However, these RNN-based data-driven constitutive models may not satisfy the underlying thermodynamics principles of path-dependent materials.

2.3 Physics-Constrained Model-Free Data Driven Computing

Another strategy in data-driven materials modeling is to bypass the constitutive modeling step by formulating an optimization problem to search for the physically admissible state that satisfies equilibrium and compatibility and minimizes the distance to a material dataset [2, 28, 94]. In this data-driven approach, the search of material data at each integration point from the material dataset is determined via a distance-minimization function and is called the distance-minimizing data-driven (DMDD) computing. This data-driven computing paradigm has been extended to dynamics [95], problems with geometrical nonlinearity [47, 96], inelasticity [97], anisotropy [5], material identification and constitutive manifold construction [98–101]. A variational framework for data-driven computing was proposed in [3, 102] to allow versatile in the employment of special approximation functions and numerical methods.

To better handle noise induced by outliers and intrinsic randomness in the experimental datasets, data-driven computing integrated with statistical models or machine learning techniques were developed, including incorporating maximum entropy estimation with a cluster analysis [103], regularization based on data sampling statistics [104], and locally convex reconstruction inspired from manifold learning for nonlinear dimensionality reduction [3]. In the local convexity data-driven (LCDD) computing proposed in [3], the convexity condition is imposed on the reconstructed material graph to avoid convergence issues that usually arise in standard data fitting approaches. The LCDD framework has been extended to model nonlinear elastic solids and applied to model mechanical responses of biological heart valve tissues with experimental data sets by [47]. Recently, [105] introduced a tensor voting machine learning technique into the entropy based DMDD framework to construct locally linear tangent spaces in order to utilize underlying data structures to achieve high-order convergence of data-driven solutions, referred to as a second-order data-driven scheme. The aforementioned physics-constrained data-driven computing frameworks have demonstrated promising performances in various scientific fields. Despite these advances, these data-driven computing frameworks cannot handle anisotropic material systems with various anisotropic orientations, i.e., orientations of material anisotropy, effectively. This is due to the fact that the data-driven solvers of these frameworks do not consider information of anisotropic orientations of materials, which prevents applications of these data-driven computing frameworks to complex material systems with directional dependent material behaviors, e.g., musculoskeletal systems with significant differences in the muscle fiber direction leading to variations in anisotropy of muscle tissue.

The above mentioned (distance-minimizing) data-driven computing approaches are distinct from the DNN-based constitutive modeling approaches. The latter constructs the surrogate models of constitutive laws independent to the solution procedure of boundary value problems, whereas the former is *model-free* and it incorporates the material data search into the solution procedure of boundary value problems. Thus, the former approach was also called *model-free data-driven* computing [97, 103]. The model-free data-driven approach circumvents the need of using material tangent during solution iteration processes, which offers another unique feature in computational mechanics. From the perspective of machine learning algorithms, the DNN approach is considered as supervised learning, and it requires pre-defined input-output functions, e.g. strain-stress laws. It has been shown that selecting the response function by DNNs within the constitutive framework is non-trivial [84, 88, 106]. On the other hand, the data-driven computing approaches with unsupervised learning such as clustering and manifold learning are capable of discovering the underlying data structure for the constitutive manifold [3, 28]. However, this type of data-driven approaches could encounter difficulties in high dimensional applications when material data sampling involves multidimensional and history-dependent state variables. As demonstrated in [2, 3, 94], higher data dimension results in lower convergence rate with respect to data size, and thus demands effective dimensionality reduction for improved effectiveness of data-driven computing. Further, the model-free data-driven schemes rely on the direct search of nearest neighbors from the material dataset, leading to limited extrapolative generalization when the distribution of data points becomes sparse.

2.4 Physics-Informed Data-Driven Reduced-Order Modeling

Physical simulations have played an increasingly significant role in developments of engineering, science, and technology. The widespread applications of physical simulations in digital twins systems [107, 108] is one recent example. Many physical processes are mathematically modeled by time-dependent nonlinear partial differential equations (PDEs). As it is difficult or even impossible to obtain analytical solutions for many highly complicated problems, various numerical methods have been developed to approximate the analytical solutions. However, due to the complexity and the domain size of problems, high-fidelity forward physical simulations can be computationally intractable even with high performance computing, which prohibits their applications to problems that require a large number of forward simulations, such as design optimization [29, 30], optimal control [31], uncertainty quantification [32, 33], and inverse analysis [33, 34].

To achieve accurate and efficient physical simulations, data can play a key role. For example, various physics-constrained data-driven model reduction techniques have been developed, such as the projection-based reduced-order model (ROM), in which the state fields of the full-order model (FOM) are projected to a linear or nonlinear subspace so that the dimension of the state fields is significantly reduced. Popular *linear* projection techniques include the proper orthogonal decomposition (POD) [35], the reduced basis method [36], and the balanced truncation method [37], while autoencoders [44, 45] are often applied for *nonlinear* projection [60, 109, 110]. The linear-subspace ROM (LS-ROM) has been successfully applied to various problems, such as nonlinear heat conduction [111], Lagrangian hydrodynamics [112–114], nonlinear diffusion equations [111, 115], Burgers equations [114, 116–118], convection-diffusion equations [119, 120], Navier-Stokes equations [121, 122], Boltzmann transport problems [123, 124], fracture mechanics [125, 126], molecular dynamics [127, 128], fatigue analysis under cycling-induced plastic deformations [59], topology optimization [129, 130], structural design optimization [131, 132], etc. Despite successes of the classical LS-ROM in many applications, it is limited to the assumption that intrinsic solution space falls into a low-dimensional subspace, which means the solution space has a small Kolmogorov n-width. This assumption is not satisfied in advection-dominated systems with sharp gradients, moving shock fronts, and turbulence, which prohibits the applications of the LS-ROM approaches for these systems. On the other hand, it has been shown that nonlinear-subspace ROMs based on autoencoders outperforms the LS-ROM on advection-dominated systems [9, 60].

Several strategies have been developed to extend LS-ROM for addressing the challenge posed by advection-dominated systems, which can be mainly categorized into Lagrangian-based approaches and methods based on a transport-invariant coordinate frame. In the Lagrangian-based approaches, Lagrangian coordinate grids are leveraged to build a ROM that propagates both the wave physics and the coordinate grid in time [133–135]. Although these methods work well, their applicability is limited by the requirement of full knowledge of the governing equations for obtaining the Lagrangian grid. The second category of strategies are based on transforming the system dynamics to a moving coordinate frame by adding a time-dependent shift to the spatial coordinates such that the system dynamics are absent of advection. Many methods have been proposed to numerically obtain the shift function for the moving frame, including the shifted POD method [136] that detects the shift by either tracking solution peaks or an expansive singular value decomposition algorithm (SVD), where different candidate shifts are applied to the data before performing SVD, and the implicit feature tracking algorithm [137] based on a minimal-residual ROM. Despite the effectiveness of these methods, the high computational costs prohibits their applications in practice. Recently, Papacicco, et al. [138] proposed a fully data-driven approach based on two separate neural networks (NNs), one to detect nonlinear shift in the transport velocity and the other one to interpolate a shifted solution back to the reference frame. However, this approach does not propose a predictive ROM to integrate the shift detection with a projection framework. Other approaches that may not fit precisely into these two categories include the work by [139] that updates the ROM basis online and the work by [140] that applies a representation of transnational features via advection modes and the subsequent residual (not purely advective features) via global modes to tackle the advection-diffusion systems.

Most physics-constrained data-driven projection-based ROMs aforementioned are *intrusive*, which require plugging the reduced-order solution representation into the discretized system of governing equations. Although the intrusive brings many benefits, such as extrapolation robustness, requirement of less training data, high accuracy, the implementation of the intrusive

ROMs requires not only sufficient understanding of the numerical solver of the high-fidelity simulation, but also access to the source code of the numerical solver.

In contrast, *non-intrusive* ROMs are purely data-driven. It requires neither access to the source code nor the knowledge of the high-fidelity solver. Many non-intrusive ROMs are constructed based on interpolation techniques that provide nonlinear mapping to relate inputs to outputs. Among various interpolation techniques, such as Gaussian processes [141, 142], radial basis functions [143, 144], Kriging [145, 146], NNs have been most popular due to their strong flexibility and capability supported by the universal approximation theorem [67]. NN-based surrogates have been applied to various physical simulations, such as fluid dynamics [147], particle simulations [148], bioinformatics [149], deep Koopman dynamical models [150], porous media flow [51, 61, 151, 152], etc. However, pure black-box NN-based surrogates lack *interpretability* and suffer from unstable and inaccurate generalization performance, especially when the training data is limited. For example, Swischuk, et al. [153] compared various ML models, including NNs, multivariate polynomial regression, *k*-nearest neighbors (KNNs), and decision trees, for learning nonlinear mapping between input parameters and low-dimensional representations of solution fields obtained by POD projection. Given input parameters unseen during training, the trained ML models are used to predict low-dimensional solution fields, which are then projected back to the high-dimensional physical space by the POD basis. The numerical examples of this study show that the flexible NN-based model performs worst, highlighting the importance of choosing an appropriate ML strategy by considering the bias-variance trade off, especially when the training data coverage of the input space is sparse.

In recent years, several ROM methods have been integrated with latent-space learning algorithms. Kim, et al. [154] proposed a DeepFluids framework in which the autoencoder was applied for nonlinear projection and a latent-space time integrator was used to approximate the evolution of the solutions in the latent space. Xie, et al. [57] applied the POD for linear projection and a multi-step NN to propagate the latent-space dynamical solutions. Hoang, et al. [155] applied the POD to compress space-time solution space to obtain space-time reduced-order basis and examined several surrogate models to map input parameters to space-time basis coefficients, including multivariate polynomial regression, KNNs, random forest, and NNs. Kadeethum, et al. [61] compared performance of the POD and autoencoder compression along with various latent space interpolation techniques, such as radial basis function and artificial neural networks. However, the latent-space dynamics models of these methods are complex and lack interpretability.

To improve the interpretability and generalization capability, it is critical to identify the underlying equations governing the latent-space dynamics. Many methods exist for identification of interpretable governing laws from data, including symbolic regression that searches both parameters and the governing equations simultaneously [156, 157] and parametric models that fit parameters to equations of a given form, such as the sparse identification of nonlinear dynamics (SINDy) [52] and operator inference [158–160]. Cranmer et al. [53] applied graph neural networks to learn sparse latent representations and symbolic regression with a genetic algorithm to discover explicit analytical relations of the learned latent representations, which enhances efficiency of symbolic regression to high-dimensional data [161]. Instead of genetic algorithms, other techniques have been applied to guide the equation search in symbolic regression, such as gradient descent [162, 163] and Monte Carlo Tree Search with asymptotic constraints in NNs [164].

Champion, et al. [165] applied an autoencoder for nonlinear projection and SINDy to identify simple ordinary differential equations (ODEs) that govern the latent-space dynamics. The autoencoder and the SINDy model were trained interactively to achieve simple latent-space dynamics. However, the proposed SINDy-autoencoder method is not parameterized and generalizable. Bai and Peng [58] proposed parametric non-intrusive ROMs that combine the POD-based linear projection with regression surrogates to approximate dynamical systems of latent variables, including support vector machines with kernel functions, tree-based methods, KNNs, vectorial kernel orthogonal greedy algorithm (VKOGA), and SINDy. The ROMs integrated with VKOGA and SINDy deliver superior cost versus error trade-off. Additionally, many non-intrusive ROMs have been developed based on POD-based linear projection with latent space dynamics captured by polynomials through operator inference [158–160, 166–175]. For example, Qian, et al. [159] introduced a lifting map to transform non-polynomial physical dynamics to quadratic polynomial dynamics and then combined POD-based linear projection with operator inference to identify quadratic reduced models for dynamical systems. Due to the limitation of the POD-based linear projection, these non-intrusive ROMs have difficulties with advection-dominated problems. To address this challenge, Issan and Kramer [176] recently proposed a non-intrusive ROM based on shifted operator inference by transforming the original coordinate frame of dynamical systems to a moving coordinate frame in which the dynamics are absent of translation and rotation. Fries, et al. [9] proposed a parametric latent space dynamics identification (LaSDI) framework in which an autoencoder was applied for nonlinear projection and a set of local dynamics identification (DI) models were introduced to identify local latent-space dynamics, as illustrated in Fig. 2.5. The LaSDI framework can be viewed as a generalization of aforementioned non-intrusive ROMs built upon latent-space dynamics identification, since it allows linear or nonlinear projection and enables latent-space dynamics to be captured by flexible DI models based on general nonlinear functions. However, since a sequential training procedure was adopted for the autoencoder and the DI models, the lack of interaction between the autoencoder and the DI models leads to strong dependency of the complexity and quality of the latent-space dynamics on the autoencoder architecture, which could pose challenges to the subsequent training of the DI models and thus affect the model performances. Most importantly, all the above-mentioned approaches rely on predefined training samples, such as uniform or Latin hypercube sampling that may not be optimal in terms of the number of samples for achieving the best model performance in the prescribed parameter space. As the generation of the simulation data can be computationally expensive, it is important to minimize the number of samples.

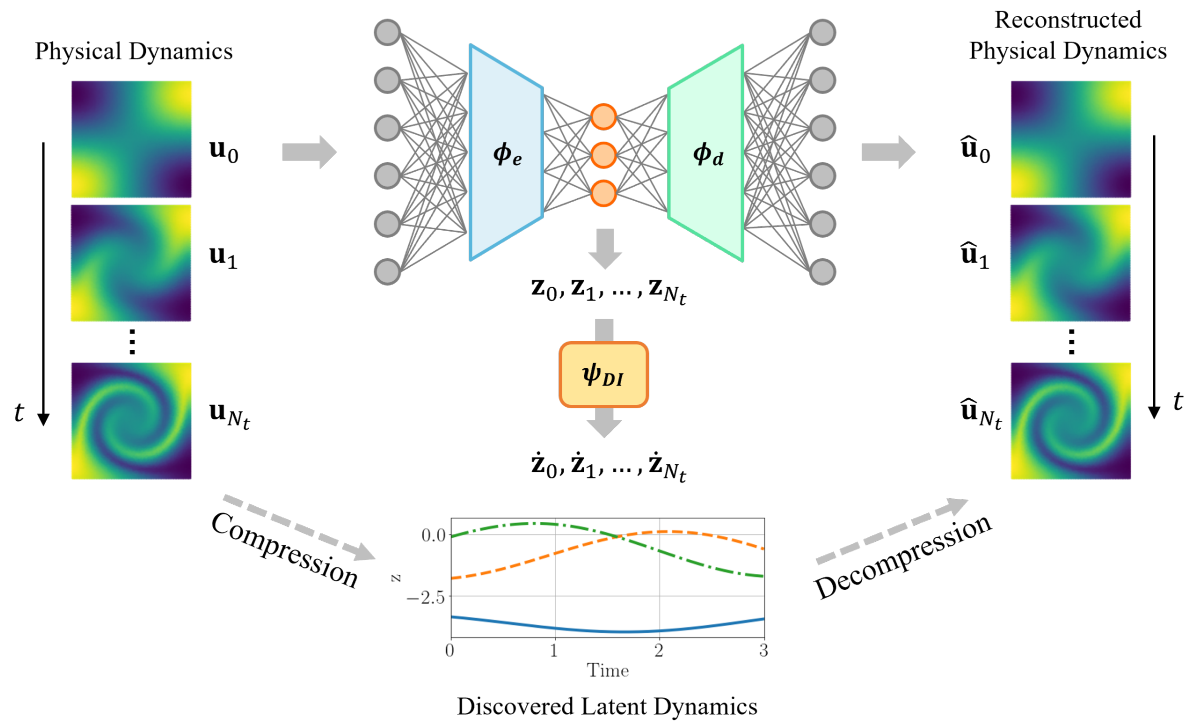


Figure 2.5. Schematics of data-driven reduced-order modeling by identification of latent-space dynamics. The autoencoder performs nonlinear projection and discovers intrinsic latent representations of high-fidelity solutions, while the dynamics identification (DI) model with strong interpretability approximate the ordinary differential equations that govern the latent-space dynamics.

Chapter 3

Thermodynamics

As the fundamental laws of physics and natural sciences, thermodynamics laws underlie natural processes of *reversible* or *irreversible* mechanical systems. The thermodynamics first law describes conservation of energy of thermodynamic systems. The thermodynamics second law deals with the directionality of thermodynamic processes. This Chapter reviews the basics of thermodynamics.

3.1 First law of thermodynamics

The first law of thermodynamics states that the total energy of a thermodynamic system and its surroundings is conserved. Mechanical and thermal energy transferred to the system (and lost by the surrounding medium) is retained in the system as part of its *total energy* consisting of *kinetic energy* associated with motion of the system's particles and *potential energy* associated with deformation. That means energy can change form, but its amount is conserved. The first law of thermodynamics can be expressed as [177]

$$\Delta E^{total} = \Delta W^{ext} + \Delta Q, \quad \text{with} \quad \Delta E^{total} = \Delta K + \Delta E^{int}, \tag{3.1}$$

where ΔE^{total} is the change of total energy; ΔK is the change of the total kinetic energy; ΔE^{int} is the change of the total internal energy; ΔW^{ext} is the change of the total external work; ΔQ is the change of total external thermal energy transferred to the system.

3.2 Second law of thermodynamics

Although the first law of thermodynamics describes the conservation of energy during thermodynamics processes, it does not provide any information about the *direction* of such processes. The direction of physical processes can be expressed as a constraint on the way *entropy* can change during any process, which is what the second law of thermodynamics is about: The entropy (S) of an isolated system can only increase or stay the same in any process [177],

$$\Delta S \geq 0. \tag{3.2}$$

If an isolated system undergoes a process with increasing entropy, then the reverse process can never occur and such a process is *irreversible*. However, if the system's entropy remains unchanged, then the reverse process is also possible and such process is *reversible*. If a process is reversible, it must also be quasistatic.

3.3 Continuum thermodynamics

3.3.1 Local form of the first law

Let us consider a continuous medium with a domain Ω and a boundary Γ. The total kinetic energy of the continuum is

$$K = \int_{\Omega} \frac{1}{2} \rho ||\mathbf{v}||^2 d\Omega, \tag{3.3}$$

where ρ and $\mathbf{v}$ denote the material density and the velocity of the infinitesimal volume element $d\Omega$, respectively. The total internal energy of the continuum is

$$E^{int} = \int_{\Omega} \rho e d\Omega, \tag{3.4}$$

where e is the specific internal energy, i.e., internal energy per unit mass.

The first law of thermodynamics in Eq. 3.1 can be expressed in its rate form as

$$\dot{K}+\dot{E}^{int}=\dot{W}^{ext}+\dot{Q}, \tag{3.5}$$

where the rates of change of the total kinetic energy and the total internal energy are respectively given by

$$\dot{K}=\frac{D}{Dt}\int_{\Omega}\frac{1}{2}\rho v_i v_i d\Omega=\int_{\Omega}\rho a_i v_i d\Omega, \tag{3.6}$$

$$\dot{E}^{int}=\frac{D}{Dt}\int_{\Omega}\rho e d\Omega=\int_{\Omega}\rho\dot{e}d\Omega, \tag{3.7}$$

where the superposed "." and $\frac{D(\cdot)}{Dt}$ denote the material time derivative; a_i denotes the acceleration. The rate of change of the total external work, $\dot{W}^{ext}$, can be expressed as

$$\dot{W}^{ext}=\int_{\Omega}\rho b_i v_i d\Omega+\int_{\Gamma}\bar{t}_i v_i d\Gamma, \tag{3.8}$$

where b_i is the body force per unit mass and $\bar{t}_i$ is the external traction acting on the surfaces of the continuum body. Considering the Cauchy's relation, $t_i=\sigma_{ij}n_j$, where σ_{ij} is the Cauchy stress, and the divergence theorem, the second term in Eq. (3.8) can be rearranged as

$$\int_{\Gamma}\bar{t}_i v_i d\Gamma=\int_{\Gamma}(\sigma_{ij}n_j)v_i d\Gamma=\int_{\Omega}(\sigma_{ij}v_i)_{,j}d\Omega=\int_{\Omega}\sigma_{ij,j}v_i+\sigma_{ij}v_{i,j}d\Omega. \tag{3.9}$$

Substituting Eq. (3.9) into Eq. (3.8) gives

$$\dot{W}^{ext}=\int_{\Omega}(\sigma_{ij,j}+\rho b_i)v_i d\Omega+\int_{\Omega}\sigma_{ij}v_{i,j}d\Omega. \tag{3.10}$$

Considering the balance of linear momentum, $\sigma_{ij,j}+\rho b_i=\rho a_i$, Eq. (3.10) becomes

$$\dot{W}^{ext}=\int_{\Omega}\rho a_i v_i d\Omega+\int_{\Omega}\sigma_{ij}v_{i,j}d\Omega=\dot{K}+\int_{\Omega}\sigma:\dot{\varepsilon}d\Omega, \tag{3.11}$$

where $\dot{\varepsilon}$ is the rate of the strain tensor and $\sigma : \dot{\varepsilon}$ denotes the rate of the mechanical work. Substituting Eq. (3.11) into the first law of thermodynamics in Eq. (3.5) gives

$$\dot{E}^{int} = \int_{\Omega} \sigma : \dot{\varepsilon} d\Omega + \dot{Q}. \tag{3.12}$$

The rate of heat transfer $\dot{Q}$ can be divided into two parts:

$$\dot{Q} = \int_{\Omega} \rho h d\Omega - \int_{\Gamma} \mathbf{q} \cdot \mathbf{n} d\Gamma, \tag{3.13}$$

where h is the specific rate of heat supply and $\mathbf{q}$ is the heat flux. Applying the divergence theorem to the second term on the right-hand side of Eq. (3.13) gives

$$\dot{Q} = \int_{\Omega} \rho h d\Omega - \int_{\Omega} \text{div } \mathbf{q} d\Omega, \tag{3.14}$$

Substituting Eqs. (3.7) and(3.14) into Eq. (3.12) gives [177, 178]

$$\begin{aligned} &\int_{\Omega} \rho \dot{e} d\Omega = \int_{\Omega} \sigma : \dot{\varepsilon} d\Omega + \int_{\Omega} \rho h d\Omega - \int_{\Omega} \text{div } \mathbf{q} d\Omega \\ &\int_{\Omega} \rho \dot{e} - \sigma : \dot{\varepsilon} - \rho h + \text{div } \mathbf{q} d\Omega = 0. \end{aligned} \tag{3.15}$$

This can be rewritten for any arbitrary subbody $\Omega_s \subseteq \Omega$, so it must be satisfied pointwise, leading to the local form of the first law of thermodynamics:

$$\rho \dot{e} = \sigma : \dot{\varepsilon} - \text{div } \mathbf{q} + \rho h. \tag{3.16}$$

3.3.2 Local form of the second law

The entropy of an arbitrary subbody Ω_s is

$$S(\Omega_s) = \int_{\Omega_s} \rho s d\Omega, \tag{3.17}$$

where s is the specific entropy, i.e., the entropy per unit mass. The Clausius-Planck inequality in its rate form is expressed as [177]

$$\dot{S} \geq \dot{S}^{ext} = \frac{\dot{Q}}{T^{RHS}}, \tag{3.18}$$

where T^{RHS} is the temperature of the reversible heat source (RHS) from which the heat is quasistatically transferred to the body. In continuum thermodynamics theory, it is assumed that the boundary points are always in thermal equilibrium with their reversible heat sources. Accordingly, we can substitute Eq. (3.13) into Eq. (3.18) and take the factor of $\frac{1}{T}$ inside the integrals where the temperature (T) is treated as a function of position:

$$\dot{S}^{ext}(\Omega_s) = \int_{\Omega_s} \frac{\rho h}{T} d\Omega - \int_{\Gamma_s} \frac{\mathbf{q}\cdot\mathbf{n}}{T} d\Gamma, \tag{3.19}$$

Substituting Eqs. (3.17) and (3.19) into Eq. (3.18) gives

$$\frac{D}{Dt}\int_{\Omega_s} \rho s d\Omega \geq \int_{\Omega_s} \frac{\rho h}{T} d\Omega - \int_{\Gamma_s} \frac{\mathbf{q}\cdot\mathbf{n}}{T} d\Gamma. \tag{3.20}$$

Applying the Reynolds transport theorem to the left-hand side and the divergence theorem to the surface integral on the right-hand side gives

$$\begin{aligned} &\int_{\Omega_s} \rho \dot{s} d\Omega \geq \int_{\Omega_s} \frac{\rho h}{T} d\Omega - \int_{\Omega_s} \text{div}\,\left(\frac{\mathbf{q}}{T}\right) d\Omega, \\ &\int_{\Omega_s} \rho \dot{s} + \text{div}\,\left(\frac{\mathbf{q}}{T}\right) - \frac{\rho h}{T} d\Omega \geq 0, \end{aligned} \tag{3.21}$$

which must hold for any arbitrary subbody Ω_s. We obtain the local form of the second law of thermodynamics:

$$\rho \dot{s} + \text{div}\,\left(\frac{\mathbf{q}}{T}\right) - \frac{\rho h}{T} \geq 0, \tag{3.22}$$

which is called the Clausius-Duhem inequality [177, 178].

3.3.3 Thermodynamic relations

Combining the first and second thermodynamic principles (Eqs. (3.16) and (3.22)) yields the dissipation inequality

$$\sigma : \dot{\varepsilon} - \rho \dot{e} + T \rho \dot{s} - \mathbf{q} \cdot \frac{\nabla T}{T} \geq 0. \tag{3.23}$$

The left-hand side of Eq. (3.23) represents the total dissipation rate that can be decomposed into the non-negative mechanical dissipation rate D and the non-negative thermal dissipation rate D^{th} [178, 179]:

$$D = \sigma : \dot{\varepsilon} - \rho \dot{e} + T \rho \dot{s} \geq 0, \tag{3.24a}$$

$$D^{th} = -\mathbf{q} \cdot \frac{\nabla T}{T} \geq 0. \tag{3.24b}$$

The equality holds only for *reversible* processes.

Considering a constant material density and defining the specific internal energy per unit volume as $E = \rho e$ and the specific entropy per unit volume as $S = \rho s$, we have $\dot{E} = \rho \dot{e}$ and $\dot{S} = \rho \dot{s}$. Therefore, Eq. (3.24a) can be rewritten as

$$D = \sigma : \dot{\varepsilon} - \dot{E} + T \dot{S} \geq 0. \tag{3.25}$$

Denoting the specific Helmholtz free energy as $F = E - TS$ [178] and taking the time derivative gives

$$\dot{F} = \dot{E} - \dot{T} S - T \dot{S}. \tag{3.26}$$

Combining Eq. (3.25) and Eq. (3.26) gives

$$\dot{F} = \sigma : \dot{\varepsilon} - D - \dot{T} S. \tag{3.27}$$

For strain-rate independent materials, the Helmholtz free energy can be defined as [178]

$$F := F(T, \varepsilon, \mathbf{z}), \tag{3.28}$$

where $\mathbf{z} = (z_1, ..., z_N)$ is a collection of N internal state variables (ISVs) introduced to characterize the state of path-dependent materials, which can also be interpreted as history variables [97]. However, ISVs are often non-measurable and the identification of the ISVs is often based on empiricism, which is non-trivial for materials with highly complex and nonlinear path-dependent behaviors. Here, a ML-enhanced data-driven approach is proposed to automatically infer the essential ISVs that follow the thermodynamics principles, which will be discussed in Chapter 7. Differentiation of Eq. (3.28) gives

$$\dot{F} = \frac{\partial F}{\partial T}\dot{T} + \frac{\partial F}{\partial \varepsilon} : \dot{\varepsilon} + \frac{\partial F}{\partial \mathbf{z}} \cdot \dot{\mathbf{z}}. \tag{3.29}$$

Equating Eq. (3.27) with Eq. (3.29) gives

$$\left(\frac{\partial F}{\partial T} + S\right)\dot{T} + \left(\frac{\partial F}{\partial \varepsilon} - \sigma\right) : \dot{\varepsilon} + \left(\frac{\partial F}{\partial \mathbf{z}} \cdot \dot{\mathbf{z}} + D\right) = 0. \tag{3.30}$$

The arbitrariness of $\dot{T}$, $\dot{\varepsilon}$, and $\dot{z}$ leads to the following relations

$$S = -\frac{\partial F}{\partial T}, \tag{3.31a}$$

$$\sigma = \frac{\partial F}{\partial \varepsilon}, \tag{3.31b}$$

$$D = -\frac{\partial F}{\partial \mathbf{z}} \cdot \dot{\mathbf{z}}. \tag{3.31c}$$

These relations are derived based on the universal thermodynamics principals. In Chapter 7, we will introduce a thermodynamically consistent machine-learned ISV approach for data-driven modeling of path-dependent materials with the consideration of the thermodynamic relations

(Eq. (3.31)).

3.3.4 Isothermal processes

An isothermal process is a thermodynamic process occurring at a constant temperature. For isothermal processes, Eq. (3.27) is reduced to

$$\dot{F} = \sigma : \dot{\varepsilon} - D. \tag{3.32}$$

The Helmholtz free energy defined in Eq. (3.28) is modified as

$$F := F(\varepsilon, \mathbf{z}). \tag{3.33}$$

Differentiation of Eq. (3.33) gives

$$\dot{F} = \frac{\partial F}{\partial \varepsilon} : \dot{\varepsilon} + \frac{\partial F}{\partial \mathbf{z}} \cdot \dot{\mathbf{z}}. \tag{3.34}$$

Equating Eq. (3.32) with Eq. (3.34) gives

$$\left(\frac{\partial F}{\partial \varepsilon} - \sigma \right) : \dot{\varepsilon} + \left(\frac{\partial F}{\partial \mathbf{z}} \cdot \dot{\mathbf{z}} + D \right) = 0. \tag{3.35}$$

The arbitrariness of $\dot{\varepsilon}$ and $\dot{z}$ leads to the following thermodynamic relations

$$\sigma = \frac{\partial F}{\partial \varepsilon}, \tag{3.36a}$$

$$D = -\frac{\partial F}{\partial \mathbf{z}} \cdot \dot{\mathbf{z}}. \tag{3.36b}$$

3.3.5 Nonlinear elasticity

A nonlinear elastic system undergoes *reversible* thermodynamic processes, which means $D^{th} = 0$ and $D = 0$. Therefore, Eq. (3.32) is further reduced to

$$\dot{F} = \sigma : \dot{\varepsilon}. \tag{3.37}$$

The Helmholtz free energy defined in Eq. (3.28) is modified as

$$F := F(\varepsilon). \tag{3.38}$$

Differentiation of Eq. (3.38) gives

$$\dot{F} = \frac{\partial F}{\partial \varepsilon} : \dot{\varepsilon}. \tag{3.39}$$

Equating Eq. (3.37) with Eq. (3.39) gives

$$\left(\frac{\partial F}{\partial \varepsilon} - \sigma \right) : \dot{\varepsilon} = 0. \tag{3.40}$$

The arbitrariness of $\dot{\varepsilon}$ leads to the following thermodynamic relation

$$\sigma = \frac{\partial F}{\partial \varepsilon}. \tag{3.41}$$

3.4 Acknowledgement

This chapter, in part, is a reprint of the material as it appears in: "Xiaolong He and Jiun-Shyan Chen. Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, page 115348, 2022". The dissertation author is the primary investigator and author of this paper.

Chapter 4

Physics-Constrained Data-Driven Computing

This Chapter introduces basic equations of the physics-constrained data-driven computing framework for nonlinear solids [3, 47, 96], followed by a review of two material data-driven local solvers and the associated computational approaches.

4.1 Governing equations of nonlinear mechanics

The equations governing the deformation of a solid in a domain Ω^X bounded by a Neumann boundary Γ_t^X and a Dirichlet boundary Γ_u^X in the undeformed configuration are given as

$$\begin{cases} DIV(\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) + \mathbf{b} = \mathbf{0}, & \text{in } \Omega^X, \\ \mathbf{E} = \mathbf{E}(\mathbf{u}) = (\mathbf{F}^T\mathbf{F} - \mathbf{I})/2, & \text{in } \Omega^X, \\ (\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) \cdot \mathbf{N} = \mathbf{t}, & \text{on } \Gamma_t^X, \\ \mathbf{u} = \mathbf{g}, & \text{on } \Gamma_u^X, \end{cases} \tag{4.1}$$

where $\mathbf{u}$ is the displacement vector, $\mathbf{E}$ is the Green Lagrangian strain tensor, $\mathbf{S}$ is the second Piola-Kirchhoff (2nd-PK) stress tensor, and DIV denotes the divergence operator. Without loss of generality, the governing equations (4.1) are defined in the reference (undeformed) configuration [180], which is denoted by the superscript "X". In Eq. (4.1), $\mathbf{F}$ is the deformation gradient related to $\mathbf{u}$, defined as $\mathbf{F}(\mathbf{u}) = \partial(\mathbf{X}+\mathbf{u})/\partial\mathbf{X}$, where $\mathbf{X}$ is the material coordinate, and $\mathbf{b}$, $\mathbf{N}$, $\mathbf{t}$, and $\mathbf{g}$ are the body force, the surface normal on Γ_t^X, the traction on Γ_t^X, and the prescribed displacement on Γ_u^X, respectively.

The first equation in (4.1) is the equilibrium. The second equation in (4.1) is the compatibility. The third and forth equations in (4.1) are the Neumann and Dirichlet boundary conditions, respectively. To obtain the solutions to the boundary value problem in Eq. (4.1), material laws that describe the relation between stress and strain are required, e.g.,

$$\mathbf{S} = \mathbf{f}(\mathbf{E}), \quad \text{in } \Omega^X. \tag{4.2}$$

The material law is typically constructed with a pre-defined function $\mathbf{f}$ based on experimental observation, mechanics principles, and mathematical simplification with model parameters calibrated from limited material data [68, 181] or by computational homogenization approaches such as FE^2 [182, 183], which inevitably introduce materials modeling empiricism and errors [184]. Moreover, the consistent tangent stiffness associated with the material law is often required in nonlinear computation [180].

For complex material systems, phenomenological material models are difficult to construct. The physics-constrained data-driven computing framework [2, 3, 28] offers an alternative to directly utilize material data and bypasses the need of phenomenological model construction.

4.2 Data-driven modeling of nonlinear elasticity

In this framework, the material behavior is described by means of strain and stress tensors $(\hat{\mathbf{E}}, \hat{\mathbf{S}})$ given by the material genome database. A material database $\mathbb{E} = \{(\hat{\mathbf{E}}_I, \hat{\mathbf{S}}_I)\}_{I=1}^M \in \mathscr{E}$ is defined to store the material data, where M is the number of material data points, and the hat symbol "$\wedge$" is used to denote material data. Here, $\mathscr{E}$ denotes the admissible set of material database, which will be further discussed in Section 4.3. To search for the most suitable (closest) strain-stress pairs $(\hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*)$ for a given state $(\mathbf{E}, \mathbf{S})$, an energy-like distance function extended from [2] is defined:

$$\mathscr{F}(\mathbf{E},\mathbf{S};\hat{\mathbf{E}}^*,\hat{\mathbf{S}}^*) = \min_{(\hat{\mathbf{E}},\hat{\mathbf{S}})\in\mathscr{E}} \int_{\Omega^X} \left(d_E^2(\mathbf{E},\hat{\mathbf{E}}) + d_S^2(\mathbf{S},\hat{\mathbf{S}})\right) d\Omega, \tag{4.3}$$

with

$$d_E^2(\mathbf{E},\hat{\mathbf{E}}) = \frac{1}{2}(\mathbf{E}-\hat{\mathbf{E}}) : \hat{\mathbb{C}} : (\mathbf{E}-\hat{\mathbf{E}}), \tag{4.4}$$

$$d_S^2(\mathbf{S},\hat{\mathbf{S}}) = \frac{1}{2}(\mathbf{S}-\hat{\mathbf{S}}) : \hat{\mathbb{C}}^{-1} : (\mathbf{S}-\hat{\mathbf{S}}), \tag{4.5}$$

where $\hat{\mathbb{C}}$ is a predefined symmetric and positive-definite tensor used to properly regulate the distances between $(\mathbf{E},\mathbf{S})$ and $(\hat{\mathbf{E}},\hat{\mathbf{S}})$. Usually, the selection of the coefficient matrix $\hat{\mathbb{C}}$ depends on the a priori knowledge of the given dataset. One widely adopted normalization scheme in machine learning is based on the variance of the data, but the selection is not unique. For example, the Mahalanobis distance of data is employed to compute the coefficient matrices [104]. Recently, He et al. [47] proposed to use the ratio of the standard deviations of the associated components of the stress–strain data to construct a diagonal coefficient matrix. Henceforth, the strain-stress pair $(\hat{\mathbf{E}},\hat{\mathbf{S}}) \in \mathscr{E}$ extracted from the material database is called the *material data (state)*, whereas $(\mathbf{E},\mathbf{S})$ is called the *physical state* if it satisfies the physically admissible set $\mathscr{C}$ given by the equilibrium and compatibility equations in Eq. (4.1), denoted as $(\mathbf{E},\mathbf{S}) \in \mathscr{C}$.

As a result, the data-driven modeling problem can be formulated as:

$$\min_{(\mathbf{E},\mathbf{S})\in\mathscr{C}} \mathscr{F}(\mathbf{E},\mathbf{S};\hat{\mathbf{E}}^*,\hat{\mathbf{S}}^*) = \min_{(\mathbf{E},\mathbf{S})\in\mathscr{C}} \min_{(\hat{\mathbf{E}},\hat{\mathbf{S}})\in\mathscr{E}} \int_{\Omega^X} \left(d_E^2(\mathbf{E},\hat{\mathbf{E}}) + d_S^2(\mathbf{S},\hat{\mathbf{S}})\right) d\Omega. \tag{4.6}$$

This data-driven problem is solved by fixed-point iterations, where the minimization of $\mathscr{F}$ with respect to $(\mathbf{E},\mathbf{S})$ and $(\hat{\mathbf{E}},\hat{\mathbf{S}})$ are performed iteratively until the intersection of two sets, $\mathscr{C}$ and $\mathscr{E}$, is found within a prescribed tolerance. We denote the minimization corresponding to the material data as the *local step*, i.e. Eq. (4.3), while the one associated with the physical state as the *global step*, which will be discussed as follows.

Given the optimal material data $(\hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*)$, the global step of the data-driven problem (4.6) is expressed as the following constrained minimization problem [47]:

$$\begin{gathered}\min_{\mathbf{u},\mathbf{S}} \mathscr{F}(\mathbf{E}(\mathbf{u}), \mathbf{S}; \hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*) = \min_{\mathbf{u},\mathbf{S}} \int_{\Omega^X} \left(d_E^2(\mathbf{E}(\mathbf{u}), \hat{\mathbf{E}}^*) + d_S^2(\mathbf{S}, \hat{\mathbf{S}}^*) \right) d\Omega \\ \text{subject to: } DIV(\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) + \mathbf{b} = \mathbf{0} \text{ in } \Omega^X, \\ (\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) \cdot \mathbf{N} = \mathbf{t} \text{ on } \Gamma_t^X.\end{gathered} \tag{4.7}$$

With the Lagrange multipliers λ and η, Eq. (4.7) is transformed to the minimization of the following functional:

$$\begin{aligned}&\mathscr{F}(\mathbf{E}(\mathbf{u}), \mathbf{S}; \hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*) + \\ &\int_{\Omega^X} \lambda \cdot [DIV(\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) + \mathbf{b}] d\Omega + \int_{\Gamma_t^X} \eta \cdot [(\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) \cdot \mathbf{N} - \mathbf{t}] d\Gamma.\end{aligned} \tag{4.8}$$

The Euler-Lagrange equation of Eq. (4.8) indicates that $\eta = -\lambda$ on Γ_t^X and $\lambda = 0$ on Γ_u^X [185]. Consequently, we have

$$\begin{aligned}&\mathscr{F}(\mathbf{E}(\mathbf{u}), \mathbf{S}; \hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*) + \\ &\int_{\Omega^X} \lambda \cdot [DIV(\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) + \mathbf{b}] d\Omega - \int_{\Gamma_t^X} \lambda \cdot [(\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) \cdot \mathbf{N} - \mathbf{t}] d\Gamma.\end{aligned} \tag{4.9}$$

By means of integration by parts and the divergence theorem, Eq. (4.9) is reformulated as

$$\begin{aligned}&\mathscr{F}(\mathbf{E}(\mathbf{u}), \mathbf{S}; \hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*) - \\ &\int_{\Omega^X} [\nabla \lambda : (\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) - \lambda \cdot \mathbf{b}] d\Omega + \int_{\Gamma_t^X} \lambda \cdot \mathbf{t} d\Gamma.\end{aligned} \tag{4.10}$$

The stationary conditions of Eq. (4.10) read:

$$\delta\mathbf{u}: \int_{\Omega^X} \delta\mathbf{E}(\mathbf{u}) : \hat{\mathbb{C}} : (\mathbf{E}(\mathbf{u}) - \hat{\mathbf{E}}^*) d\Omega = \int_{\Omega^X} \delta\mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda : \mathbf{S} d\Omega, \tag{4.11a}$$

$$\delta\mathbf{S}: \int_{\Omega^X} \delta\mathbf{S} : (\hat{\mathbb{C}}^{-1} : \mathbf{S} - \mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda) d\Omega = \int_{\Omega^X} \delta\mathbf{S} : \hat{\mathbb{C}}^{-1} : \hat{\mathbf{S}}^* d\Omega, \tag{4.11b}$$

$$\delta\lambda: \int_{\Omega^X} \delta\nabla\lambda : (\mathbf{F}(\mathbf{u}) \cdot \mathbf{S}) d\Omega = \int_{\Omega^X} \delta\lambda \cdot \mathbf{b} d\Omega + \int_{\Gamma_t^X} \delta\lambda \cdot \mathbf{t} d\Gamma. \tag{4.11c}$$

As Eq. (4.11b) provides correction between the physical stress $\mathbf{S}$ and the material stress data $\mathbf{S}^*$, a collocation approach is considered in Eq. (4.11b) to yield:

$$\mathbf{S} = \hat{\mathbb{C}} : (\mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda) + \hat{\mathbf{S}}^*, \tag{4.12}$$

which represents the stress solution update. Substituting Eq. (4.12) into Eqs. (4.11a) and (4.11c) yields:

$$\begin{aligned} \int_{\Omega^X} \left[\delta\mathbf{E}(\mathbf{u}) : \hat{\mathbb{C}} : (\mathbf{E}(\mathbf{u}) - \hat{\mathbf{E}}^*) - (\delta\mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda) : \hat{\mathbb{C}} : (\mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda)\right] d\Omega \\ = \int_{\Omega^X} (\delta\mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda) : \hat{\mathbf{S}}^* d\Omega, \end{aligned} \tag{4.13a}$$

$$\begin{aligned} \int_{\Omega^X} (\mathbf{F}^T(\mathbf{u}) \cdot \delta\nabla\lambda) : [\hat{\mathbb{C}} : (\mathbf{F}^T(\mathbf{u}) \cdot \nabla\lambda) + \hat{\mathbf{S}}^*] d\Omega \\ = \int_{\Omega^X} \delta\lambda \cdot \mathbf{b} d\Omega + \int_{\Gamma_t^X} \delta\lambda \cdot \mathbf{t} d\Gamma. \end{aligned} \tag{4.13b}$$

The solutions $\mathbf{u}$ and λ are solved from Eqs. (4.13) by means of the Newton-Raphson method [180], and the physical state stress $\mathbf{S}$ is subsequently obtained from (4.12). As such, Eqs. (4.12)-(4.13) are the computational procedures to solve Eq. (4.7). Moreover, in this boundary value problem, Eq. (4.13), the optimal material data $(\hat{\mathbf{E}}^*, \hat{\mathbf{S}}^*)$ not only provides the underlying material information learned from material database, but also serves as the material data-based connection to relate the strain in compatibility and the stress in equilibrium equations in Eqs. (4.13a) and (4.13b), respectively.

In this study, the reproducing kernel particle method (RKPM) [186, 187] is employed to discretize the unknown fields u and λ in Eqs. (4.13) due to its capabilities of nodal approximation of state variables and enhanced smoothness that are particularly effective for data-driven computing. The formulation of the reproducing kernel approximation is given in Section 4.4. Moreover, for effectiveness in data-driven computing, a stabilized nodal integration scheme (SCNI [188], see Section 4.5) is used to integrate the weak formulations in Eq. (4.13) for reducing the number of integration points where the stress and strain material data need to be searched [3].

Combining the ease of introducing arbitrary order of continuity in the RK approximation as well as the employment of the SCNI scheme to integrate the weak equations in Eq. (4.13), it allows the material data search and variable evaluation to be performed only at the nodal points, avoiding the necessity of computing field and state variables separately at nodal points and Gauss points, respectively, for enhanced efficiency and accuracy of data-driven computing. In addition, under the SCNI framework, the nodal physical stress $\mathbf{S}$ is directly associated with the material stress data without introducing additional interpolation errors, see [3, 47] for more details. The Green Lagrangian strain tensor $\mathbf{E}$ is computed from the RK approximated displacement $\mathbf{u}$ and is evaluated at the nodal points. Note that stress update equation in Eq. (4.12) is also carried out nodally.

The physical state $(\mathbf{E},\mathbf{S})$ evaluated at the integration points $\mathbf{x}_\alpha$ are denoted as $\{(\mathbf{E}_\alpha,\mathbf{S}_\alpha)\}_{\alpha=1}^{N}$, where N is the number of integration points. Note that due to the employment of nodal integration [3, 188], the integration points share the same set of points as the nodal points. For simplicity of notation, we denote $\mathbf{z}_\alpha = (\mathbf{E}_\alpha,\mathbf{S}_\alpha)$ as the physics state and $\hat{\mathbf{z}}_\alpha = (\hat{\mathbf{E}}_\alpha,\hat{\mathbf{S}}_\alpha)$ the material data associated with the integration point α in the following discussions.

In data-driven computing, the local step (4.3) and the global step (4.7) are solved iteratively to search for the optimal material data from the admissible material set $\mathscr{E}$ that is closest to the physical state satisfying the physical constraints given in Eq. (4.1). The convergence properties of this fixed-point iteration solver have been investigated in [2, 94]. It should be noted that the selection of the optimal material data by the material data-driven local solver is crucial to the effectiveness of data-driven computing [3, 103], and further discussion will be presented in the following Sections 4.3 and 6.3.

4.3 Material data-driven local solver

4.3.1 Distance-minimizing data-driven (DMDD) local solver

The effectiveness of data-driven computational paradigm discussed in Section 4.2 relies heavily on the search of optimal material data $\hat{\mathbf{z}}^*_\alpha = (\hat{\mathbf{E}}^*_\alpha, \hat{\mathbf{S}}^*_\alpha)$, $\alpha = 1, ..., N$, from the material dataset $\mathbb{E}$. When adopting the distance-minimizing data-driven (DMDD) approach proposed in [2], the local material solver in Eq. (4.3) used to find the optimal material data can be defined as

$$\begin{aligned} (\hat{\mathbf{E}}^*_\alpha, \hat{\mathbf{S}}^*_\alpha) &= \underset{(\hat{\mathbf{E}}_\alpha, \hat{\mathbf{S}}_\alpha) \in \mathbb{E}}{\arg\min}\; d_E^2(\mathbf{E}_\alpha, \hat{\mathbf{E}}_\alpha) + d_S^2(\mathbf{S}_\alpha, \hat{\mathbf{S}}_\alpha), \quad \alpha = 1, ..., N, \\ \hat{\mathbf{z}}^*_\alpha &= \underset{\hat{\mathbf{z}}_\alpha \in \mathbb{E}}{\arg\min}\; d_z^2(\mathbf{z}_\alpha, \hat{\mathbf{z}}_\alpha), \quad \alpha = 1, ..., N, \end{aligned} \tag{4.14}$$

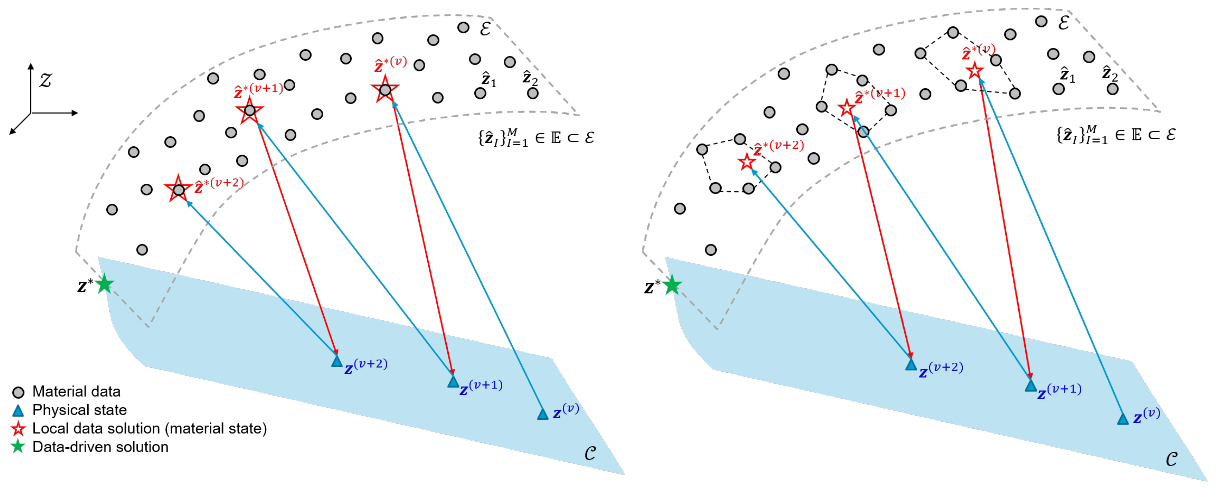
with

$$d_z^2(\mathbf{z}, \hat{\mathbf{z}}) = d_E^2(\mathbf{E}, \hat{\mathbf{E}}) + d_S^2(\mathbf{S}, \hat{\mathbf{S}}), \tag{4.15}$$

where the distance functions d_E and d_S are referred to Eqs. (4.4) and (4.5), α denotes the indices of integration points, and N is the total number of integration points. The direct searching of the closest material data can lead to inaccurate data-driven solutions when the material data contains noise and outliers.

4.3.2 Local convexity-preserving data-driven (LCDD) local solver

For enhanced data-driven computing, especially with sparse noisy data, an alternative approach called local convexity data-driven (LCDD) computing was proposed in [3] by introducing the underlying structure of material data via manifold learning. In this approach, the local



(a) DMDD **(b)** LCDD

Figure 4.1. Geometric schematics of the (a) DMDD [2] and (b) LCDD [3] solvers. The data-driven solution $\mathbf{z}^*$ is given by the intersection of the admissible set of physical states $\mathscr{C}$ and the material admissible set $\mathscr{E}$.

solver is expressed as:

$$
\begin{aligned}
(\hat{\mathbf{E}}_\alpha^*, \hat{\mathbf{S}}_\alpha^*) &= \underset{(\hat{\mathbf{E}}_\alpha, \hat{\mathbf{S}}_\alpha) \in \mathscr{E}_\alpha^l}{\arg\min} \; d_E^2(\mathbf{E}_\alpha, \hat{\mathbf{E}}_\alpha) + d_S^2(\mathbf{S}_\alpha, \hat{\mathbf{S}}_\alpha), \qquad \alpha = 1, \ldots, N, \\
\hat{\mathbf{z}}_\alpha^* &= \underset{\hat{\mathbf{z}}_\alpha \in \mathscr{E}_\alpha^l}{\arg\min} \; d_z^2(\mathbf{z}_\alpha, \hat{\mathbf{z}}_\alpha), \qquad \alpha = 1, \ldots, N,
\end{aligned}
\tag{4.16}
$$

where $\mathscr{E}_\alpha^l := \mathscr{E}^l(\mathbf{z}_\alpha)$ denotes a local convex subset formed by k material data points closest to the given physical state $\mathbf{z}_\alpha = (\mathbf{E}_\alpha, \mathbf{S}_\alpha)$, providing a smooth and bounded solution space for optimal material data search, and preserving the convexity of the constructed local material manifold for enhanced robustness and convergence stability. The material step defined in Eq. (4.16) involves two substeps. First, given a physical state $\mathbf{z}_\alpha$, k nearest neighbors (material data points), $\{\hat{\mathbf{z}}_i\}_{i \in \mathscr{N}_k(\mathbf{z}_\alpha)} \subset \mathbb{E}$, are identified based on the distance measured by $d_z^2(\mathbf{z}_\alpha, \hat{\mathbf{z}})$, where the indices of the nearest neighbors of $\mathbf{z}_\alpha$ are stored in $\mathscr{N}_k(\mathbf{z}_\alpha)$. Then, a local convex space is constructed based on the collected nearest neighbors $\{\hat{\mathbf{z}}_i\}_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)}$ of $\mathbf{z}_\alpha$, defined as

$$\begin{aligned}\mathscr{E}^l(\mathbf{z}_\alpha) &= \mathrm{Conv}\left(\{\hat{\mathbf{z}}_i\}_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)}\right)\\ &= \left\{\sum_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)} w_i\hat{\mathbf{z}}_i \;\middle|\; \sum_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)} w_i = 1 \text{ and } w_i \geq 0,\ \forall i \in \mathcal{N}_k(\mathbf{z}_\alpha)\right\}.\end{aligned} \tag{4.17}$$

The optimal coefficients $\mathbf{w}_\alpha = \{w_i\}_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)}$ are obtained by solving the following minimization problem:

$$\begin{aligned}\mathbf{w}^*_\alpha = \arg\min_{\mathbf{w}_\alpha} d_z^2\left(\mathbf{z}_\alpha, \sum_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)} w_i\hat{\mathbf{z}}_i\right)\\ \text{subject to: } \sum_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)} w_i = 1,\\ w_i \geq 0,\ \forall i \in \mathcal{N}_k(\mathbf{z}_\alpha),\end{aligned} \tag{4.18}$$

where $\mathbf{w}^*_\alpha = \{w^*_i\}_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)}$ denotes the optimal coefficients. Eq. (4.18) is solved by means of a non-negative least-square algorithm with penalty relaxation, see details in [38]. The optimal material data $\hat{\mathbf{z}}^*_\alpha$ can then be obtained by the following local convex construction:

$$\hat{\mathbf{z}}^*_\alpha = \sum_{i\in\mathcal{N}_k(\mathbf{z}_\alpha)} w^*_i\hat{\mathbf{z}}_i, \tag{4.19}$$

which ensures that the optimal material data $\hat{\mathbf{z}}^*_\alpha$ always lie within the local convex space $\mathscr{E}^l_\alpha$. The LCDD computing process is depicted in Fig. 4.1(b), where the material solver finds the optimal material data within the local convex space $\mathscr{E}^l_\alpha$ (denoted by enclosed black dash lines) formed by the k selected data points closest to the given physical state, which expands the feasible solution space for robustness in data-driven iterations against noise and outliers in the material data [38].

Fig. 4.1 shows the comparison of the DMDD and LCDD solvers, where (v) is the iteration index and one iteration consists of solving one global (physical) step, i.e. Eqs. (4.12)–(4.13) and one material data-driven local step, e.g. Eq. (4.14) or (4.16), as noted in Section 4.2. The local step of the DMDD solver (4.14) searches for the material data closest to the given physical state directly from the material dataset $\mathbb{E}$, see Fig. 4.1(a). It has been shown that this heuristic solver suffers from noisy dataset and requires enormous data to guarantee satisfactory accuracy [3, 103]. On the other hand, the LCDD solver (4.16) searches for the optimal material data based on the locally constructed convex space $\mathscr{E}_\alpha^l$ informed by the neighboring data, as shown in Fig. 4.1(b). The key idea behind the construction of $\mathscr{E}_\alpha^l$ is to provide a smooth, bounded and lower dimensional admissible space for optimal material data search in Eq. (4.16), and to preserve the convexity of the constructed local material manifold for enhanced robustness and stability in data-driven iterations.

While the material data-driven local solver in Eq. (4.16) locates the optimal data from the defined feasible set (constructed by a set of local neighboring points), the final solution of the associated data-driven modeling problem Eq. (4.6) is not guaranteed to be globally optimal. This is consistent to other existing data-driven approaches [2, 3, 94] where the optimality fundamentally depends on the characteristic of the material dataset. However, as demonstrated in references [2, 3, 94], if the material dataset is well posed, the proposed data-driven solver can converge optimally as the density of data points increases.

Remark. *It should be noticed that in both Eqs. (4.14) and (4.16) the nearest points are sought based on the metric functions d_E and d_S. Thus, it suffers from the notorious "dimensionality curse" when data-driven modeling attempts to scale up to high-dimensional material data. Although the innate manifold learning in LCDD allows noise and dimensionality reduction, the proper definition of the metric functions in high-dimensional phase space remains challenging [27]. Besides, as the nearest neighbors are searched locally from the existing data points of the material dataset, it leads to limited extrapolative generalization to be demonstrated in Section 6.4.2. Furthermore, the data search and the locally convex reconstruction through a constrained minimization solver at every local step during data-driven computation could result in high computational cost especially for the large and high dimensional material dataset.*

4.4 Reproducing kernel approximation

The reproducing kernel (RK) approximations [186, 187] is adopted in the physical solver of the data-driven computing framework due to its nodal approximation of state and field variables that are particularly effective for data-driven computing. The RK approximation functions can be constructed to possess desired completeness and continuity, which are determined by basis functions and kernel functions, respectively. Fig. 4.2(a) shows a domain Ω discretized by a set of nodes.

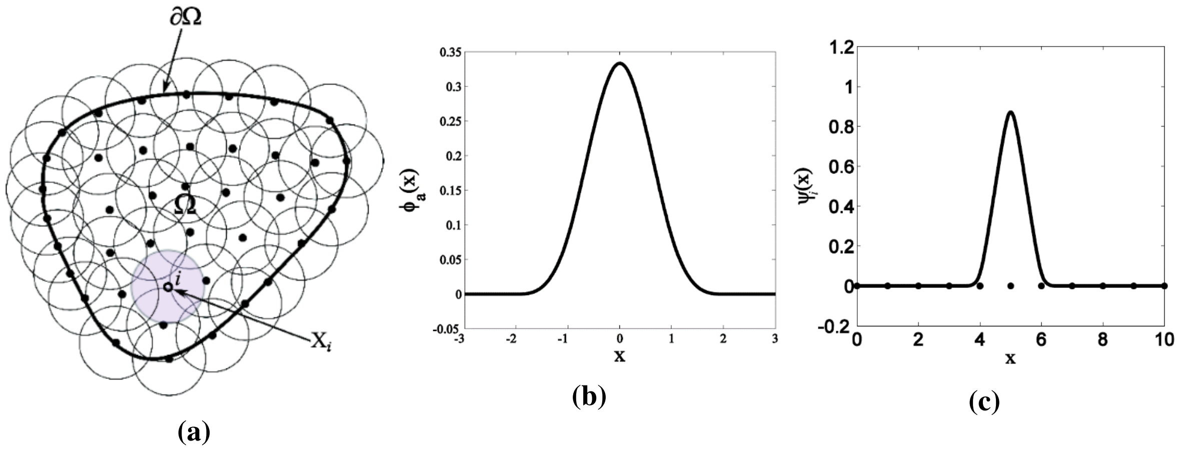


Figure 4.2. (a) A domain Ω discretized by the a set of RK nodes; (b) a cubic B-spline function widely used as a kernel function in RK approximation; (c) an example of RK approximation function centered at $X = 5$ with a support size $a = 1.5 \times$ (nodal spacing)

The displacement field $\mathbf{u}(\mathbf{x})$ and the Lagrange multiplier $\lambda(\mathbf{x})$ in weak-form equations Eq. (4.13) are approximated by

$$\mathbf{u}^h(\mathbf{x}) = \sum_{I=1}^{NP} \Psi_I(\mathbf{x})\mathbf{d}_I, \tag{4.20a}$$

$$\lambda^h(\mathbf{x}) = \sum_{I=1}^{NP} \Psi_I(\mathbf{x})\Lambda_I, \tag{4.20b}$$

where $\mathbf{d}_I$ and Λ_I are the nodal coefficients associated with the fields $\mathbf{u}(\mathbf{x})$ and $\lambda(\mathbf{x})$, respectively, and $\Psi_I(\mathbf{x})$ is the reproducing kernel (RK) approximation function expressed as

$$\Psi_I(\mathbf{x}) = \mathbf{H}^T(\mathbf{x}-\mathbf{x}_I)\mathbf{b}(\mathbf{x})\phi_a(\mathbf{x}-\mathbf{x}_I), \tag{4.21}$$

where $\mathbf{H}^T(\mathbf{x}-\mathbf{x}_I) = [1, x_1 - x_{1I}, x_2 - x_{2I}, x_3 - x_{3I}, ..., (x_3 - x_{3I})^n]$ is a vector of monomial basis functions up to the n-th order, and $\phi_a(\mathbf{x}-\mathbf{x}_I)$ is a kernel function with a local support size "a", controlling the smoothness of the RK approximation function, for example, the cubic B-spline kernel function, see Fig. 4.2(b):

$$\phi_a(y) = \begin{cases} \frac{2}{3} - 4y^2 + 4y^3, & 0 \le y < \frac{1}{2} \\ \frac{4}{3} - 4y + 4y^2 - \frac{4}{3}y^3, & \frac{1}{2} \le y < 1 \\ 0, & y \ge 1 \end{cases} \quad \text{with } y = \frac{||\mathbf{x}-\mathbf{x}_I||}{a}. \tag{4.22}$$

In Eq. (4.21), $\mathbf{b}(\mathbf{x})$ is a parameter vector determined by imposing the n-th order reproducing conditions [186, 187],

$$\sum_{I=1}^{NP} \Psi_I(\mathbf{x}) x_{1I}^i x_{2I}^j x_{3I}^k = x_1^i x_2^j x_3^k, \quad |i+j+k| = 0, 1, ..., n. \tag{4.23}$$

Substituting Eq. (4.21) into Eq. (4.23) yields $\mathbf{b}(\mathbf{x}) = \mathbf{M}^{-1}(\mathbf{x})\mathbf{H}(\mathbf{0})$, where $\mathbf{M}(\mathbf{x})$ is a moment matrix given by

$$\mathbf{M}(\mathbf{x}) = \sum_{I=1}^{NP} \mathbf{H}(\mathbf{x}-\mathbf{x}_I)\mathbf{H}^T(\mathbf{x}-\mathbf{x}_I)\phi_a(\mathbf{x}-\mathbf{x}_I). \tag{4.24}$$

The RK approximation function is then obtained as, see Fig. 4.2(c):

$$\Psi_I(\mathbf{x}) = \mathbf{H}^T(\mathbf{0})\mathbf{M}^{-1}(\mathbf{x})\mathbf{H}(\mathbf{x}-\mathbf{x}_I)\phi_a(\mathbf{x}-\mathbf{x}_I). \tag{4.25}$$

4.5 Nodal integration scheme

The stabilized conforming nodal integration (SCNI) approach is employed for the domain integration of the weak form (Eq. (4.13)) to achieve computational efficiency and accuracy when using RK shape functions with nodal integration quadrature schemes.

The key idea behind SCNI is to satisfy the linear patch test (thus, ensure the linear consistency) by leveraging a condition, i.e. the divergence constraint on the test function space and numerical integration [188], expressed as:

$$\hat{\int}_{\Omega} \nabla\Psi_I d\Omega = \hat{\int}_{\partial\Omega} \Psi_I \mathbf{n} d\Gamma, \tag{4.26}$$

where 'ˆ' over the integral symbol denotes numerical integration. In SCNI, an effective way to achieve Eq. (4.26) is based on nodal integration with gradients smoothed over conforming representative nodal domains, as shown in Fig. 4.3, converted to boundary integration using the divergence theorem

$$\tilde{\nabla}\Psi_I(\mathbf{x}_L) = \frac{1}{V_L}\int_{\Omega_L} \nabla\Psi_I d\Omega = \frac{1}{V_L}\int_{\partial\Omega_L} \Delta\Psi_I \mathbf{n} d\Gamma, \tag{4.27}$$

where $V_L = \int_{\Omega_L} d\Omega$ is the volume of a conforming smoothing domain associated with the node $\mathbf{x}_L$, and $\tilde{\nabla}$ denotes the smoothed gradient operator. In this method, smoothed gradients are employed for both test and trial functions, as the approximation in Eq. (4.27) enjoys first order completeness and leads to a quadratic rate of convergence for solving linear solid problems by meshfree Galerkin methods. As shown in Fig. 4.3, the continuum domain Ω is partitioned into N conforming cells by Voronoi diagram, and both the nodal displacement vectors and the state variables (e.g., stress, strain) are defined at the set of nodes $\{\mathbf{x}_L\}_{L=1}^{N}$.

Therefore, if we consider two-dimensional elasticity problem under the SCNI framework, the smoothed strain-displacement matrix $\tilde{\mathbf{B}}_I(\mathbf{x}_L)$ used in (16) is expressed as:

$$\tilde{\mathbf{B}}_I(\mathbf{x}_L) = \begin{bmatrix} \tilde{b}_{I1}(\mathbf{x}_L) & 0 \\ 0 & \tilde{b}_{I2}(\mathbf{x}_L) \\ \tilde{b}_{I2}(\mathbf{x}_L) & \tilde{b}_{I1}(\mathbf{x}_L) \end{bmatrix}, \tag{4.28}$$

with

$$\tilde{b}_{Ii}(\mathbf{x}_L) = \frac{1}{V_L} \int_{\partial\Omega_L} \Psi_I(\mathbf{x}) n_i(\mathbf{x}) d\Gamma. \tag{4.29}$$

Since the employment of the smoothed gradient operator in Eq. (4.27) and Eq. (4.29) satisfies the divergence constraint regardless of the numerical boundary integration, a trapezoidal rule for each segment of $\partial\Omega_L$ is used in this study.

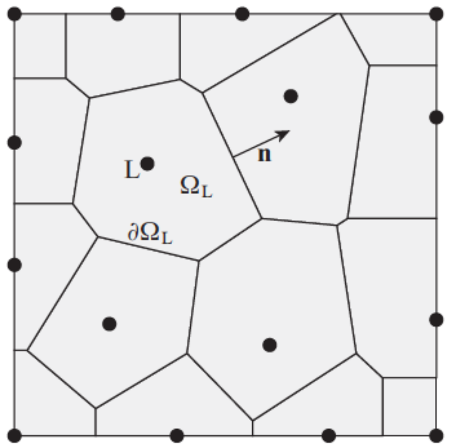


Figure 4.3. Illustration of Voronoi diagram for SCNI.

4.6 Acknowledgement

This chapter, in part, is a reprint of the materials as they appear in: “Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, and Shantanu Sinha. Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids. *Data-Centric Engineering*, 1, 2020” and ”Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021". The dissertation author is the primary investigator and author of these papers.

Chapter 5

Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids

5.1 Introduction

In recent years, the physics-constrained data-driven computing framework introduced in Chapter 4 has been extended to dynamics [95], nonlinear elasticity [47, 96], inelasticity [97], constitutive manifold construction [189, 190], and multiscale modeling [191, 192]. Despite these advances, the data-driven computing frameworks cannot handle anisotropic material systems with various anisotropic orientations, i.e., orientations of material anisotropy, effectively. This is due to the fact that the data-driven solvers of these frameworks do not consider information of anisotropic orientations of materials, which prevents applications of these data-driven computing frameworks to complex material systems with directional dependent material behaviors, e.g., musculoskeletal systems with significant differences in the muscle fiber direction leading to variations in anisotropy of muscle tissue.

The objective of this study is to develop a new data-driven solver built upon the local convexity-preserving reconstruction scheme [38] in order to model anisotropic nonlinear elastic solids. The remainder of this chapter is organized as follows. Section 5.2 introduces a new local convexity-preserving material solver designed for anisotropic solids. Particularly, a rotated material database is constructed offline and a two-level data search is integrated into the material solver to capture directional dependent material behaviors during online data-driven computing. In Section 5.3, the proposed data-driven computing framework is verified by modeling the deflection of a cantilever beam with layers containing different fiber directions and inflation of a cylinder where fiber directions vary along the circumferential direction of the cylinder. The data-driven solutions are compared with the constitutive model-based reference solutions to examine the effectiveness and robustness of the proposed methods. Concluding remarks and discussions are given in Section 5.4.

5.2 Methodologies

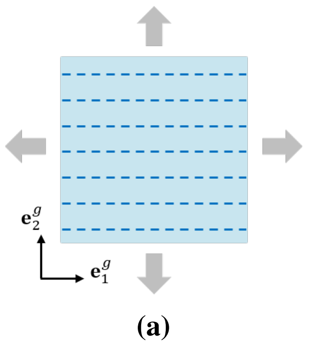
The basic formulations of physics-constrained data-driven computing framework for nonlinear elastic solids have been revisited in Chapter 4, where the distinction between the data-driven local material solvers of the DMDD [2] and LCDD [38, 47] is discussed. To model anisotropic solids, we propose to integrate a two-level data search scheme into the local convexity-preserving material solver in LCDD in order to capture the anisotropic material properties with anisotropic orientations (orientations of material anisotropy) information.

5.2.1 Local Convexity-Preserving Material Solver for Anisotropic Solids

To capture directional dependent material properties of anisotropic solids, a two-level local data search scheme is introduced in the local convexity-preserving material solver. Let us consider an anisotropic material with material behaviors characterized by a data set $\mathbb{E} = \{(\hat{\mathbf{E}}_j, \hat{\mathbf{S}}_j)\}_{j=1}^{M}$ measured in a global reference frame, and every material point in a physical system is associated with its anisotropic orientations (orientations of material anisotropy), which can be represented by the vector of Euler angles θ_i between the local fiber frame and the global reference frame, where $i = 1, ..., N$ is the index of material (integration) points. For simplicity, the methodologies of the proposed material solver for anisotropic solids is illustrated by using the examples with in-plane (two-dimensional) anisotropy, which can be easily extended to the three-dimensional problems.

Fig. 5.1(a) shows an in-plane anisotropic material sample under testing in a global reference frame $(\mathbf{e}_1^g, \mathbf{e}_2^g)$, where the dash lines indicate the material anisotropic orientation in $\mathbf{e}_1^g$ direction. A bar under uniaxial stretching, as shown in Fig. 5.1(b), contains material points $\mathbf{X}_i$ associated with certain angles (anisotropic orientations) θ_i between the local fiber frame $(\mathbf{e}_1^l, \mathbf{e}_2^l)$ of material point i and the reference frame. The corresponding rotation tensor is defined as

$$\mathbf{R}_i = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.1}$$



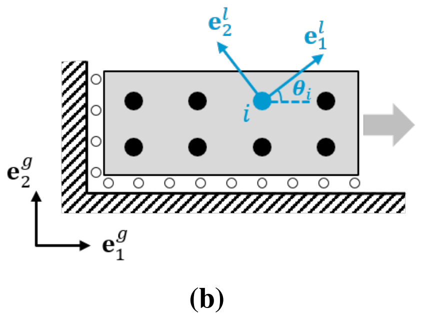


Figure 5.1. (a) Material sample under testing in a reference frame where the dash lines indicate the material anisotropic orientation; (b) uniaxial stretching of a bar. The material behaviors of the material point marked in blue are characterized by the material data from the sample shown in Fig. 5.1(a)

Applying the rotation $\mathbf{R}_i$ to the strain-stress data $\mathbb{E} = \{(\hat{\mathbf{E}}_j, \hat{\mathbf{S}}_j)\}_{j=1}^M$ yields the rotated strain-stress data $\mathbb{E}_i^\theta = \{(\hat{\mathbf{E}}_j^\theta, \hat{\mathbf{S}}_j^\theta)\}_{j=1}^M$ representing the material behaviors with the anisotropic orientation θ_i under the reference frame:

$$\hat{\mathbf{E}}_j^\theta = \mathbf{R}_i \cdot \hat{\mathbf{E}}_j \cdot \mathbf{R}_i^T, \quad j = 1, ..., M \tag{5.2a}$$

$$\hat{\mathbf{S}}_j^\theta = \mathbf{R}_i \cdot \hat{\mathbf{S}}_j \cdot \mathbf{R}_i^T, \quad j = 1, ..., M. \tag{5.2b}$$

The rotated material data sets are normally obtained offline with various angles associated with material points in the physical system. Rotated material data sets are then used to reconstruct the optimal material data in the material solver during online data-driven computing.

However, it is impractical to prepare rotated material data sets when a system contains a large number of varying anisotropic orientations associated with material points as a collection of all possible rotated material data sets requires prohibitive memory. A typical example is muscle tissues in musculoskeletal systems that involve randomly oriented fibers. To effectively model anisotropic behaviors in complex material systems, we introduce a two-level local data search scheme into the material solver. To this end, the anisotropic orientation θ is encoded as an additional feature in material data and physical states of material points. Consequently, the distance between the material data and physical state is not only measured by the distance between their strain-stress values, called *state distance*, but also the distance between their anisotropic orientations, called *anisotropic distance*, expressed as

$$d_{zf}^2\big((\mathbf{z},\theta),(\mathbf{z},\hat{\theta})\big) = d_z^2(\mathbf{z},\hat{\mathbf{z}}) + d_f^2(\theta,\hat{\theta}). \tag{5.3}$$

The state distance $d_z(\mathbf{z},\hat{\mathbf{z}})$ is computed by Eqs. (4.4) and (4.5) and the anisotropic distance $d_f(\theta,\hat{\theta})$ is defined as

$$d_f(\theta,\hat{\theta}) = ||\theta - \hat{\theta}||, \tag{5.4}$$

where $\hat{\theta}$ and θ denote the anisotropic orientations of material data and the material point, respectively.

Level-1 Data Search

Let us consider a system constituted by an anisotropic material with various anisotropic orientations, e.g., a musculoskeletal system consisting muscle fibers with various fiber orientations. The anisotropic material with various anisotropic orientations in the global reference frame exhibits the same material behaviours under the local fiber frame. We first prepare a rotated material database that contains $\hat{m}$ rotated material data sets obtained by rotating the original data set with $\hat{m}$ different angles (anisotropic orientations). Each rotated material data sets contains strain-stress data $\hat{\mathbf{z}}$ and an anisotropic orientation $\hat{\theta}$. The range of variations in anisotropic orientations of the rotated material database is sufficiently large to cover all material points in the system. Given a physical state of a material point, $(\mathbf{z}_i, \theta_i)$, where the subscript i denotes the index of a material (integration) point $\mathbf{X}_i$, we first compute the *anisotropic distance* (Eq. (5.4)) between the material point (θ_i) and all rotated material data sets ($\hat{\theta}_p$, $p = 1, 2, ..., \hat{m}$). Two rotated material data sets, $\mathbb{E}_p^\theta$ and $\mathbb{E}_q^\theta$, will then be selected so that $\hat{\theta}_p \le \theta_i \le \hat{\theta}_q$, as illustrated by the blue dash block in Fig. 5.2. The selected material data sets have the anisotropic orientations closest to that of the material point and therefore are considered to best represent the anisotropic material behaviors of the material point.

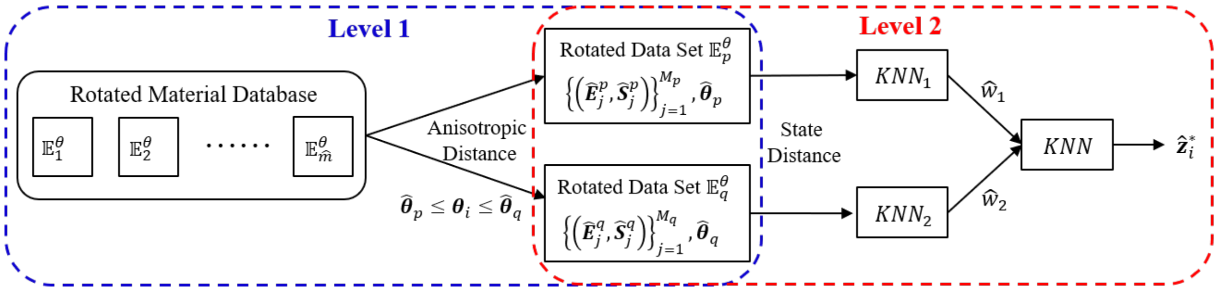

Level 1
Level 2
Rotated Material Database
$\mathbb{E}_1^\theta$
$\mathbb{E}_2^\theta$
......
$\mathbb{E}_{\hat{m}}^\theta$
Anisotropic Distance
$\hat{\theta}_p \le \theta_i \le \hat{\theta}_q$
Rotated Data Set $\mathbb{E}_p^\theta$
$\{(\hat{E}_j^p, \hat{S}_j^p)\}_{j=1}^{M_p}, \hat{\theta}_p$
Rotated Data Set $\mathbb{E}_q^\theta$
$\{(\hat{E}_j^q, \hat{S}_j^q)\}_{j=1}^{M_q}, \hat{\theta}_q$
State Distance
KNN_1
KNN_2
$\hat{w}_1$
$\hat{w}_2$
KNN
$\hat{\mathbf{z}}_i^*$

Figure 5.2. Illustration of two-level local data search in the proposed material solver

Level-2 Data Search

Given the two selected rotated material data sets by the Level-1 search scheme, we search for k nearest neighbors (KNN) within each data set based on the state distance between the physical state of the material point and the selected material data sets. The lists of k material data points closest to the physical state of the material point from the first and second selected material data set are denoted as KNN_1 and KNN_2, respectively, as shown in Fig. 5.2. To properly consider the effect of the two KNN data sets, we propose to use a linear weighting scheme based on the anisotropic orientation information as follows:

$$\hat{w}_2 = \frac{d_f(\theta_i, \hat{\theta}_p)}{d_f(\hat{\theta}_q, \hat{\theta}_p)} = \frac{||\theta_i - \hat{\theta}_p||}{||\hat{\theta}_q - \hat{\theta}_p||}, \tag{5.5a}$$

$$\hat{w}_1 = 1 - \hat{w}_2. \tag{5.5b}$$

The final list of k nearest neighbors, which is formed by int($k \cdot \hat{w}_1$) (rounded to the nearest integer) nearest neighbors from KNN_1 and int($k \cdot \hat{w}_2$) nearest neighbors from KNN_2, is used for local convexity-preserving reconstruction of the optimal material data that is closest to the physical state of the material point by

$$(\hat{\mathbf{z}}_i^*, \hat{\theta}_i^*) = \underset{(\hat{\mathbf{z}}_i, \hat{\theta}_i) \in \tilde{\mathscr{E}}(\mathbf{z}_i^*, \theta_i)}{\arg\min} \; d_z^2(\mathbf{z}_i^*, \hat{\mathbf{z}}_i) + d_f^2(\theta_i, \hat{\theta}_i), \quad i = 1, ..., N, \tag{5.6}$$

where $\mathbf{z}_i^*$ is the optimal strain-stress state obtained from the physical step (Eq. (4.7)) for the material point $\mathbf{X}_i$, θ_i is the anisotropic orientation of the material point $\mathbf{X}_i$, $\tilde{\mathscr{E}}(\mathbf{z}_i^*, \theta_i)$ is a local convex space formed by the selected k nearest neighbors $\{(\hat{\mathbf{z}}_\alpha, \hat{\theta}_\alpha)\}_{\alpha \in \mathscr{N}_k(\mathbf{z}_i^*, \theta_i)}$ of the physical state $(\mathbf{z}_i^*, \theta_i)$ based on the distance measured by $d_z^2(\mathbf{z}_i^*, \hat{\mathbf{z}}) + d_f^2(\theta_i, \hat{\theta})$. The anisotropic oritentations θ_i and $\hat{\theta}_i$ are normalized by $\hat{\theta}_q$ before the calculation of anisotropic distance.

Note that the construction of convexity-preserving local manifold takes the anisotropic orientations of the selected nearest neighbors into account in forming the local convex space $\tilde{\mathscr{E}}(\mathbf{z}_i^*, \theta_i)$ so that the anisotropic orientations of nearest neighbors play an important role in reconstructing the optimal material data $(\hat{\mathbf{z}}_i^*, \hat{\theta}_i^*)$ closest to the physical state $(\mathbf{z}_i^*, \theta_i)$. Let $\mathbf{a}_i^*$ and $\hat{\mathbf{a}}_i$ denote $(\mathbf{z}_i^*, \theta_i)$ and $(\hat{\mathbf{z}}_i, \hat{\theta}_i)$, respectively. The total distance between $\mathbf{a}_i^*$ and $\hat{\mathbf{a}}_i$ is denoted by $d_{zf}^2(\mathbf{a}_i^*, \hat{\mathbf{a}}_i)$. The local convex space $\tilde{\mathscr{E}}(\mathbf{a}_i^*)$ is constructed based on the collected nearest neighbors

$\{\hat{\mathbf{a}}_\alpha\}_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}$, defined as

$$\begin{aligned}\tilde{\mathscr{E}}(\mathbf{a}_i^*) &= \mathrm{Conv}\Big(\{\hat{\mathbf{a}}_\alpha\}_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}\Big)\\ &= \left\{\sum_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}\tilde{w}_\alpha\hat{\mathbf{a}}_\alpha \;\middle|\; \sum_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}\tilde{w}_\alpha = 1 \text{ and } \tilde{w}_\alpha \geq 0,\ \forall\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)\right\},\end{aligned} \tag{5.7}$$

The optimal coefficients $\tilde{\mathbf{w}}_i = \{\tilde{w}_\alpha\}_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}$ are obtained by solving the following minimization problem:

$$\begin{aligned}\tilde{\mathbf{w}}_i^* = \arg\min_{\tilde{\mathbf{w}}_i} d_{zf}^2\Big(\mathbf{a}_i^*, \sum_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}\tilde{w}_\alpha\hat{\mathbf{a}}_\alpha\Big)\\ \text{subject to: } \sum_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}\tilde{w}_\alpha = 1,\\ \tilde{w}_\alpha \geq 0,\ \forall\alpha\in\mathscr{N}_k(\mathbf{a}_i^*),\end{aligned} \tag{5.8}$$

where $\tilde{\mathbf{w}}_i^* = \{\tilde{w}_\alpha^*\}_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}$ denote the optimal coefficients. Eq. (5.8) is solved by the non-negative least-square algorithm with penalty relaxation used to solve for the optimal coefficients of the local convex space in LCDD's material solver (Eq. (4.18)), see more details in [38]. The optimal material data $\hat{\mathbf{a}}_i^* = (\hat{\mathbf{z}}_i^*, \hat{\boldsymbol{\theta}}_i^*)$ can then be obtained by the following local convex construction:

$$\hat{\mathbf{a}}_i^* = \sum_{\alpha\in\mathscr{N}_k(\mathbf{a}_i^*)}\tilde{w}_\alpha^*\hat{\mathbf{a}}_\alpha, \tag{5.9}$$

The optimal material data sought from the material solver is considered to be "closest" to the physical state in terms of both state distance and anisotropic distance and thus best represent the anisotropic material properties of the material point. The optimal material data will then be input to the physical step (Eq. ((4.12))-(4.13)) to solve for the closest physical state in the next data-driven iteration. Note that anisotropic properties are embedded in material database, which are independent of physical laws. Hence, the physical solver only requires the optimal strain-stress material data, $\hat{\mathbf{z}}_i^*$, reconstructed from the material solver.

Schematic Illustration

Fig. 5.3 shows schematic examples of the material step in a two-dimensional space, where $\hat{\theta}_p = 20°$, $\hat{\theta}_q = 40°$, and $k = 6$ are considered. Three different anisotropic orientations θ_i of the material point i are considered, i.e., 36°, 30° and 24°. The linear weights for selecting the nearest neighbors (*KNN* weights) from each material data set are computed based on the anisotropic orientation information by Eq. (5.5). The number of nearest neighbors from each material data set are computed by multiplying their *KNN* weights with the total number of nearest neighbors k, as listed in Table 5.1, which are rounded to the nearest integer. This approach follows the idea of instance-based learning [193] to learn the underlying relation from the neighboring manifold of observation data, but additionally, it assigns different number of votes to the selected data sets based on their anisotropic distance d_f (Eq. (5.4)). For example, in the case that the anisotropic orientation of the material point i is $\theta_i = 36°$, which is closer to that of data set 2 ($\hat{\theta}_q = 40°$) and thus the computed *KNN* weight for data set 2 is $\hat{w}_2 = 0.8$, larger than $\hat{w}_1 = 0.2$ of data set 1 ($\hat{\theta}_p = 20°$). Hence, there are 1 and 5 nearest neighbors selected from data set 1 and data set 2, respectively, which are used to form a local convex space capturing the underlying anisotropic data structure, as shown in Fig. 5.3(a). The optimal material data (red circle in 5.3(a)) is then reconstructed from the local convex space. As the anisotropic orientation of the material point is closer to that of data set 2, more nearest neighbors are selected from data set 2 for local convex reconstruction and therefore the optimal material data that is closest to the given physical state will have anisotropic properties closer to that of data set 2. In the case that the anisotropic orientation is $\theta_i = 30°$, both data sets have the same *KNN* weights and contribute the same number of nearest neighbors to the construction of local convex space, as shown in Fig. 5.3(b), indicating that they have the same effects on reconstructing the optimal material data.

Note that the relation between the anisotropic properties and anisotropic orientations is nonlinear. More sophisticated anisotropic distance metrics and local data reconstruction schemes are required in order to achieve higher accuracy in anisotropic material data reconstruction

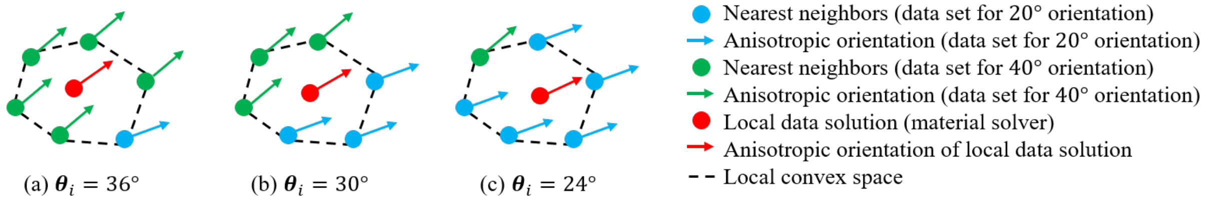

Nearest neighbors (data set for 20° orientation)
Anisotropic orientation (data set for 20° orientation)
Nearest neighbors (data set for 40° orientation)
Anisotropic orientation (data set for 40° orientation)
Local data solution (material solver)
Anisotropic orientation of local data solution
Local convex space
(a) θi = 36°
(b) θi = 30°
(c) θi = 24°

Figure 5.3. Schematic illustration of the proposed material solver for anisotropic solids. Data set 1 has an anisotropic orientation of $\hat{\theta}_p = 20^\circ$. Data set 2 has an anisotropic orientation of $\hat{\theta}_q = 40^\circ$. The total number of nearest neighbors k is 6. The material step of a material point with different anisotropic orientations are compared: (a) $\theta_i = 36^\circ$; (b) $\theta_i = 30^\circ$; (c) $\theta_i = 24^\circ$

Table 5.1. The number of nearest neighbors from each data set in the examples shown in Fig. 5.3 with $\hat{\theta}_p = 20^\circ$, $\hat{\theta}_q = 40^\circ$, and $k = 6$. The weights are computed by Eq. (5.5). $k \cdot \hat{w}_1$ and $k \cdot \hat{w}_2$ are rounded to the nearest integer.

θ_i	$\hat{w}_1$	$\hat{w}_2$	$k \cdot \hat{w}_1$	$k \cdot \hat{w}_2$
36°	0.2	0.8	1	5
30°	0.5	0.5	3	3
24°	0.8	0.2	5	1

if the selected material data sets have a large anisotropic distance, which will be investigated in our future study. In the cases that the anisotropic distance of the selected rotated material data sets is small, the L_2-based metric for anisotropic distance (Eq. (5.4)) and the linear local convexity-preserving reconstruction scheme (Eq. (5.9)) adopted in this study can yield desirable accuracy in the data-driven modeling of anisotropic materials, which will be demonstrated in the numerical examples in the following sections.

5.2.2 Local Convexity-Preserving Data-Driven Solver for Anisotropic Solids

Given an anisotropic material data set $\mathbb{E}$ and anisotropic orientations of material points in the system, the proposed data-driven solver for modeling anisotropic solids is summarized as followings.

Offline Stage:

Step 1. Define a range for variations in anisotropic orientations for material data rotation so that it covers that of the material points in the system. Depending on the distribution of the anisotropic orientations in the system, anisotropic orientations for data rotation can be evenly distributed in the defined range or follow certain statistical distributions in order to better capture anisotropic properties of the system.

Step 2. Construct a rotated material database $\mathbb{E}_{em} = \mathbb{E}_1^{\theta} \times ... \times \mathbb{E}_m^{\theta}$ by rotating the original anisotropic material data set $\mathbb{E}$ with the anisotropic orientations defined in *Step 1*. The rotation of strain and stress data is performed by Eq. (5.2).

Online Stage:

Step 1. Randomly initialize $\hat{\mathbf{z}}_i^{(0)}$, $i = 1,...,N$, where i denote the indices of material points and N is the number of material points in the system, and set the data-driven iteration index $v = 0$.

Step 2. Solve the physical step (Eq. (4.12)-(4.13)) for physical states of all material points, $\mathbf{z}_i^{*(v)}$, $i = 1,...,N$.

Step 3. For each physical state of a material point, $(\mathbf{z}_i^{*(v)}, \theta_i)$, perform two-level local data search (material step):

3.1. Search for two rotated material data sets with anisotropic orientations $\hat{\theta}_p$ and $\hat{\theta}_q$ so that $\hat{\theta}_p \leq \theta_i \leq \hat{\theta}_q$.

3.2. From each selected material data set, search for k material data points that are closest to the physical state based on the state distance $d_z^2(\mathbf{z}_i^{*(v)}, \hat{\mathbf{z}})$ and obtain the final *KNN* using the weights computed by Eq. (5.5).

3.3. Construct local convex space by solving Eq. (5.8) and obtain the optimal material data $\hat{\mathbf{z}}_i^{*(v)}$ within the local convex space by Eq. (5.9).

Step 4. Update $v = v+1$. If $\underset{i=1,...,N}{max} d_z^2(\hat{\mathbf{z}}_i^{*(v)}, \hat{\mathbf{z}}_i^{*(v-1)}) >$ tolerance, repeat *Step 2 - 4*.

Step 5. Data-driven solution: $\mathbf{z}_i^* = \hat{\mathbf{z}}_i^{*(v)}$, $i = 1,...,N$.

Remark: *Modeling anisotropic materials by classical computational mechanics requires online rotation of element stiffness matrices from local fiber frames to the global reference frame before global assembly, which could lead to high computational cost especially for large systems. In the proposed data-driven modeling of anisotropic materials, online rotation is replaced with an* ***offline rotation****, where anisotropic material data sets are rotated by various anisotropic orientations under the global reference frame to form a rotated material database for efficient online data-driven computing. In the general three-dimensional orthotropic case, each anisotropic orientation is associated with three Euler angles and thus for every anisotropic orientation considered, offline rotation operations associated with three Euler angles are required for every stress-strain data, which could be CPU intensive but can be performed in parallel especially when dealing with a large amount of data. Once the rotated material database is constructed offline, it can be efficiently applied to different material points in data-driven modeling and to different systems with the same anisotropic materials. Note that when the size of anisotropic material data sets is large and the range of variations in anisotropic orientations of the system is very broad, the offline rotated material database may require a large amount of memory.*

If memory resources are not available, one can also adopt online rotation in the data-driven computing for modeling anisotropic materials. In this case, at each data-driven iteration, the physical step remains unchanged. In the material step, the physical states are first rotated from the global reference frame to local fiber frames of material points and then the LCDD material solver (Eqs. (4.16)-(4.19)) can be applied to find the optimal material data from the unrotated anisotropic material data set. The optimal material data is then rotated back to the global reference frame for the next data-driven iteration. Compared to the proposed data-driven approach with offline rotation, this online rotation approach requires higher CPU but less memory.

5.3 Results and Discussion

5.3.1 Preparation of Material Data Sets

For the following numerical demonstration, the two-dimensional Saint Venant-Kirchhoff phenomenological model with isotropic and orthotropic elastic tensors are respectively considered as the reference models and used to generate synthetic data for data-driven computing. The plane-stress isotropic elastic stress-strain relation in the Voigt notation is expressed as

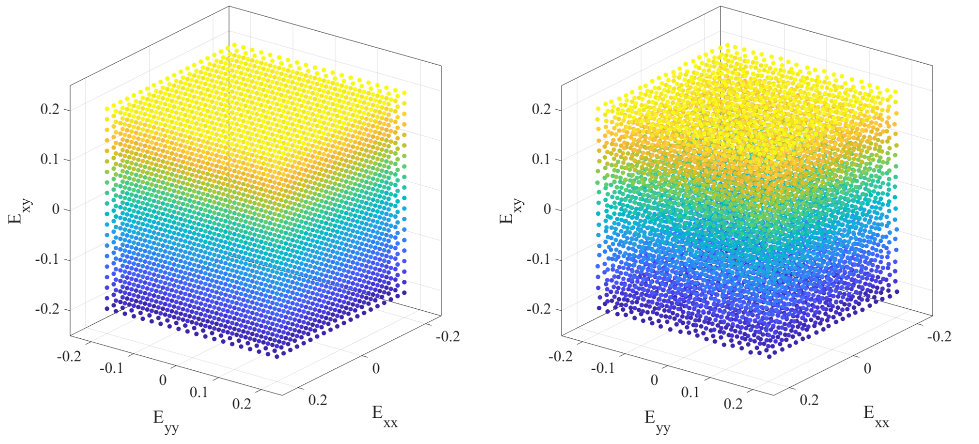
$$\begin{bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} E_{11} \\ E_{22} \\ 2E_{12} \end{bmatrix}, \tag{5.10}$$

where E denotes the Young's modulus and ν is the Poisson's ratio. As another reference material model, the plane-stress orthotropic elastic stress-strain relation in the Voigt notation is expressed as

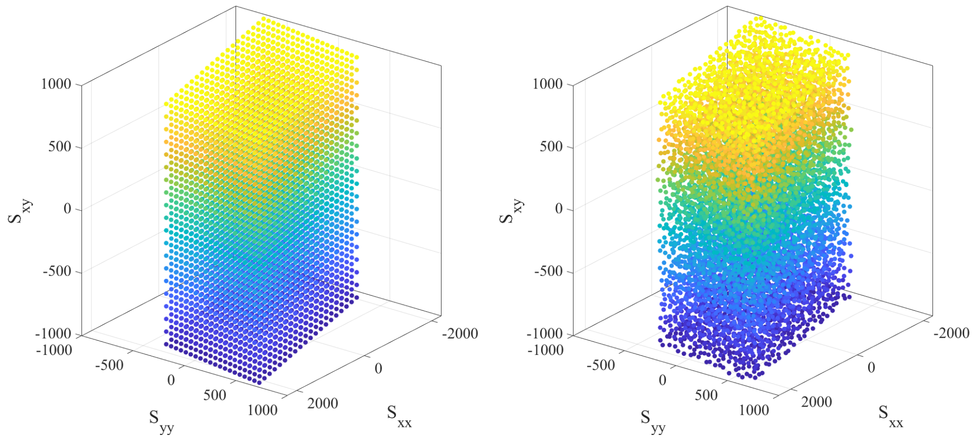
$$\begin{bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{bmatrix} = \frac{1}{1-\nu_{12}\nu_{21}} \begin{bmatrix} E_1 & \nu_{21}E_1 & 0 \\ \nu_{12}E_2 & E_2 & 0 \\ 0 & 0 & G_{12}(1-\nu_{12}\nu_{21}) \end{bmatrix} \begin{bmatrix} E_{11} \\ E_{22} \\ 2E_{12} \end{bmatrix}, \tag{5.11}$$

where E_1 and E_2 are Young's moduli in the $\mathbf{e}_1^g$ and $\mathbf{e}_2^g$ directions of the reference frame, respectively, see Fig. 5.1(a). ν_{12} and ν_{21} are Poisson's radios. G_{12} is the shear modulus.

To demonstrate the robustness of the proposed data-driven computing framework against noise presented in given material data sets, noisy material data sets are generated and the procedure is described below. First, a noiseless material data set, $\mathbb{E}^0 = \{(\hat{\mathbf{E}}_i^0, \hat{\mathbf{S}}_i^0)\}_{i=1}^M$ with $M = 20^3$, is generated, where each strain component is uniformly distributed within a certain range, e.g., [-0.2,0.2], and the stress components are obtained by using the orthotropic material model in Eq. (5.11). $M = 20^3$ strain and stress data points with $E_1 = 10^4, E_2 = 2.5 \times 10^3, \nu_{21} = 0.1, \nu_{12} = 0.4$, and $G_{12} = 4.8 \times 10^3$ are shown in Fig. 5.4(a) and (c), respectively, serving as a noiseless base data set without any noise and rotation. The anisotropic orientation of the noiseless base data set is along the horizontal direction of the reference frame, as shown in Fig. 5.1(a). Then, each component of the noiseless base data set is perturbed by Gaussian noise with a scaling factor, $0.4\bar{\mathbf{z}}_{max}/\sqrt[3]{M}$, where $\bar{\mathbf{z}}_{max}$ is a vector of the maximum values for each component among the noiseless data set. Fig. 5.4(b) and (d) show the strain and stress of the noisy base data set, $\mathbb{E} = \{(\hat{\mathbf{E}}_i, \hat{\mathbf{S}}_i)\}_{i=1}^{M}$ with $M = 20^3$, generated from the noiseless base data set $\mathbb{E}^0$ shown in Fig. 5.4(a) and (c), respectively.



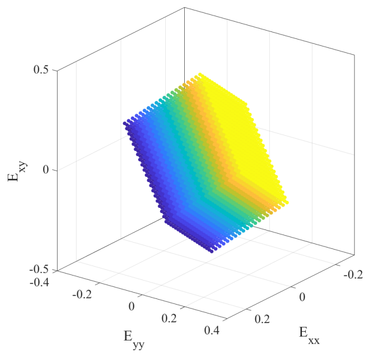
(a) Noiseless Strain Data **(b)** Noisy Strain Data



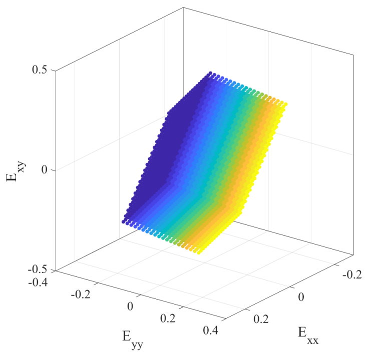
(c) Noiseless Stress Data **(d)** Noisy Stress Data

Figure 5.4. Material data sets with 8000 data points, $E_1 = 10^4, E_2 = 2.5 \times 10^3, \nu_{12} = 0.1, \nu_{21} = 0.4$, and $G_{12} = 4.8 \times 10^3$: (a) noiseless strain data; (b) noisy strain data; (c) noiseless stress data; (d) noisy stress data

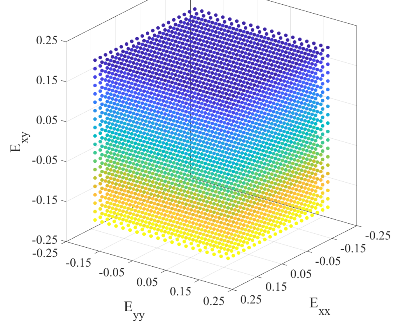
Given anisotropic orientations of material points in a physical system, e.g., θ_i of the material point *i* in Fig. 5.1(b), the strain and stress data of the base data set can be rotated using Eq. (5.2). Fig. 5.5 shows the strain and stress data rotated from the base data set by three different angles 30°, 60°, and 90°. In the following numerical examples, rotation of material data sets is performed offline by various angles (anisotropic orientations) to cover the range of variations in anisotropic orientations of systems. A collection of rotated material data sets form a rotated material database, which is then used for online data-driven computing with the material solver equipped with two-level data search designed for capturing directional dependent material behaviors.



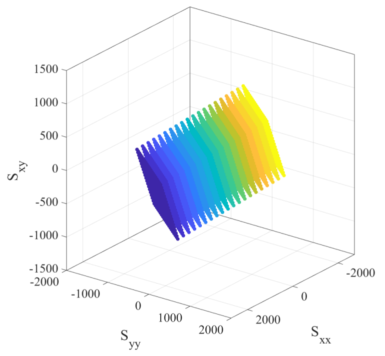
(a) Rotation: 30°



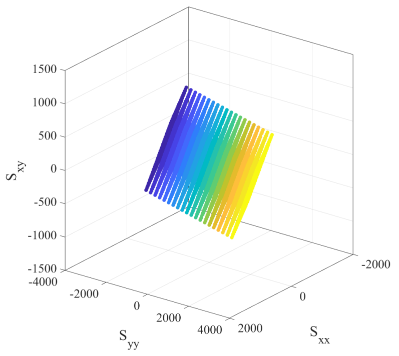
(b) Rotation: 60°



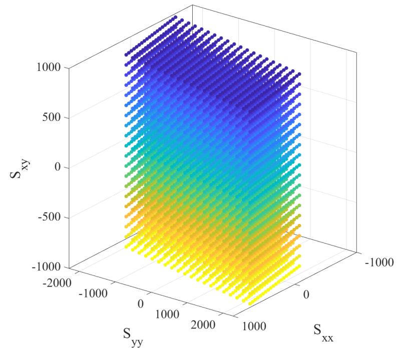
(c) Rotation: 90°



(d) Rotation: 30°



(e) Rotation: 60°



(f) Rotation: 90°

Figure 5.5. Rotated material data sets with 8000 data points, $E_1 = 10^4, E_2 = 2.5 \times 10^3, \nu_{12} = 0.1, \nu_{21} = 0.4$, and $G_{12} = 4.8 \times 10^3$: (a) strain data rotated by 30°; (b) strain data rotated by 60°; (c) strain data rotated by 90°; (d) stress data rotated by 30°; (e) stress data rotated by 60°; (f) stress data rotated by 90°

5.3.2 Multi-layer Anisotropic Cantilever Beam Subjected to A Tip Shear Load

We first investigate a multi-layer anisotropic cantilever beam benchmark problem to verify the proposed data-driven modeling framework for anisotropic nonlinear solids, where three layers of orthotropic materials are considered. Two cases are examined and compared. In Case 1, all material points in the beam have an identical anisotropic orientation, which is along the horizontal direction, as shown in Fig. 5.6(a). In Case 2, the beam is made of three layers of anisotropic materials with different anisotropic orientations. The material points within each layer have the same anisotropic orientation. Fig. 5.6(b) shows that the bottom, the middle, and the top layers of the beam have anisotropic orientations of -45°, 0°, and 45°, respectively. A synthetic noiseless orthotropic material data set with 8000 strain-stress data points is first generated from the phenomenological orthotropic elastic model with $E_1 = 10^4, E_2 = 2.5 \times 10^3, \nu_{21} = 0.1, \nu_{12} = 0.4$, $G_{12} = 4.8 \times 10^3$, and a range of [-0.2,0.2] for each strain component, as shown in Fig. 5.4(a) and (c). Then, a noisy orthotropic material data set is generated from the noiseless data set using the procedure described in Section 3.1, as shown in Fig. 5.4(b) and (d).

For Case 1, no rotation of the material data set is required as the anisotropic orientations of materials points in the beam are identical with that of the synthetic data set, which is along the horizontal direction. For Case 2, a rotated material database is constructed by rotating the generated synthetic noiseless/noisy data set from -60° to 60° with a 10° interval, i.e., -60°, -50°, -40°, -30°, -20°, -10°, 0°, 10°, 20°, 30°, 40°, 50°, and 60°. The rotated material database is then used for online data-driven computing. The numerical studies of both cases are performed with noiseless or noisy material data sets. A tip shear load $P = 10E_1I/L^2$ with $I = H^3/12$ is applied and the data-driven analysis is performed with 20 loading steps.

The normalized tip deflection-loading and stress distribution of data-driven solutions are compared with those of the constitutive model-based reference solutions obtained from an in-house Finite Element code, as shown in Figs. 5.7 and 5.8. The data-driven solutions of both cases

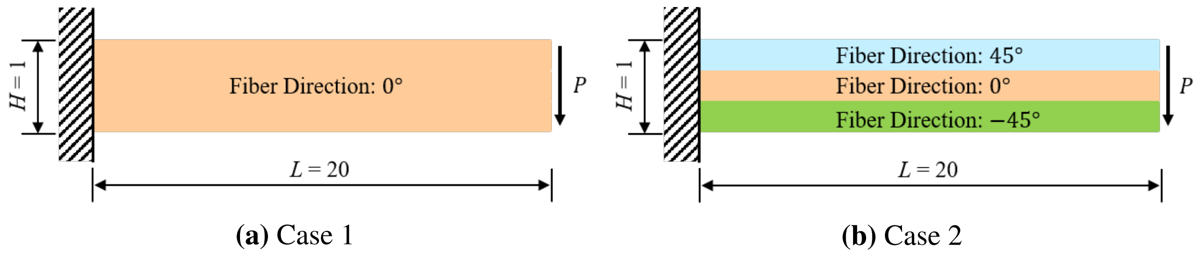


Figure 5.6. Schematic of cantilever beam subjected to a tip shear load: (a) Case 1: material points have only one anisotropic orientation 0°; (b) Case 2: material points located in different layers of the beam have different anisotropic orientations. The anisotropic orientations of the bottom, the middle, and the top layers are $-45°$, $0°$, and $45°$, respectively. $P = 10E_1I/L^2$, and $I = H^3/12$

show a satisfactory agreement with the model-based reference solutions, which demonstrates the effectiveness of the proposed data-driven computing framework for modeling anisotropic nonlinear elastic materials. The data-driven solutions obtained from using the noisy material data sets agree well with the model-based reference solutions, showing the robustness of the proposed framework against noise in the data sets, which is achieved by the local convexity-preserving reconstruction in the material solver.

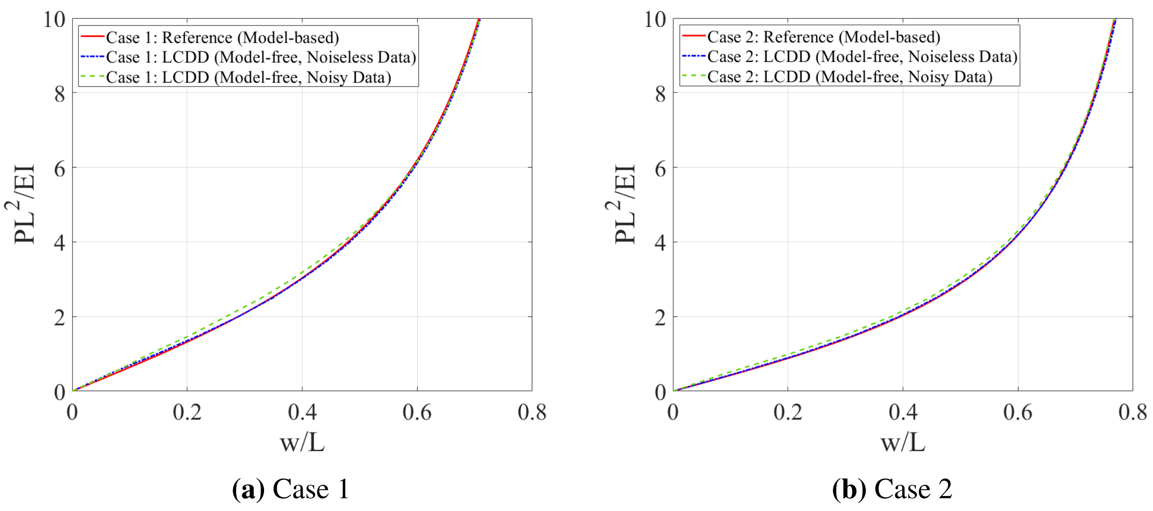
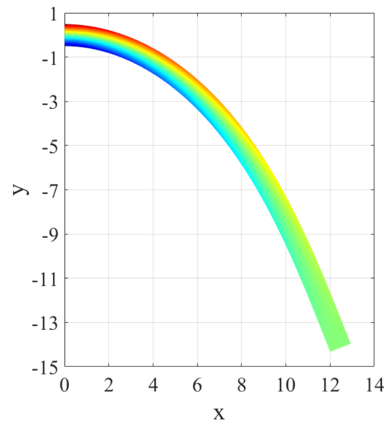
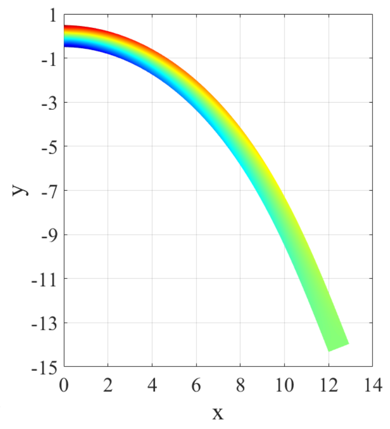


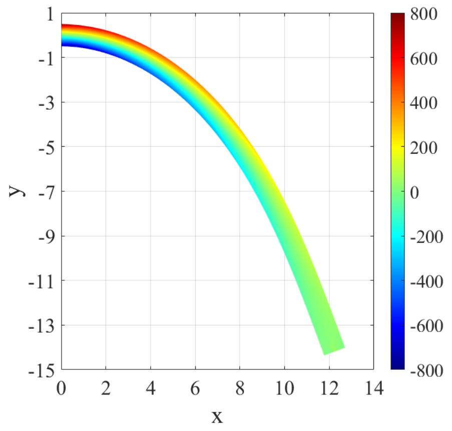
Figure 5.7. Comparison of data-driven solutions with constitutive model-based reference solutions. Normalized tip deflection-loading, where $I = H^3/12$: (a) Case 1; (b) Case 2



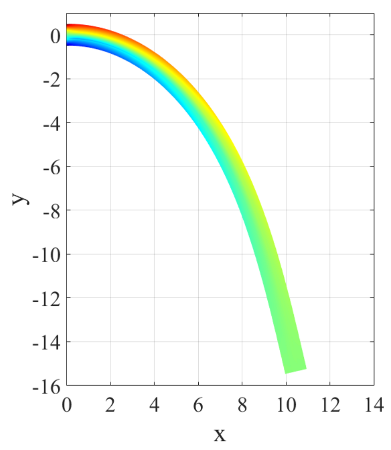
(a) Case 1 - Reference



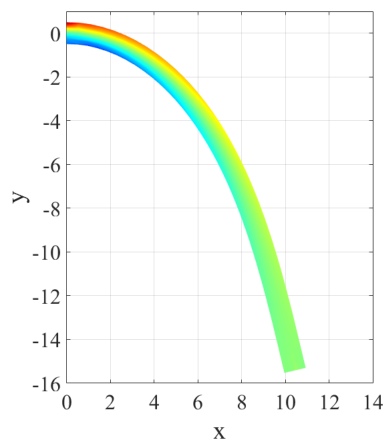
(b) Case 1 - LCDD (Noiseless data)



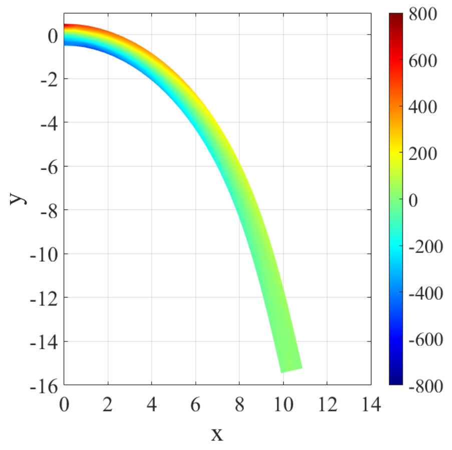
(c) Case 1 - LCDD (Noisy data)



(d) Case 2 - Reference



(e) Case 2 - LCDD (Noiseless data)



(f) Case 2 - LCDD (Noisy data)

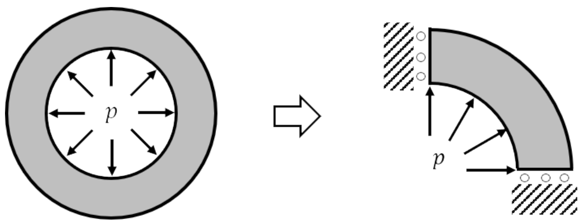
Figure 5.8. Comparison of data-driven solutions with constitutive model-based reference solutions. Distribution of S_{xx}: (a) Case 1: reference solution; (b) Case 1: data-driven solution with noiseless data; (c) Case 1: data-driven solution with noisy data; (d) Case 2: reference solution; (e) Case 2: data-driven solution with noiseless data; (f) Case 2: data-driven solution with noisy data

5.3.3 Anisotropic Cylinder Subjected to Internal Pressure

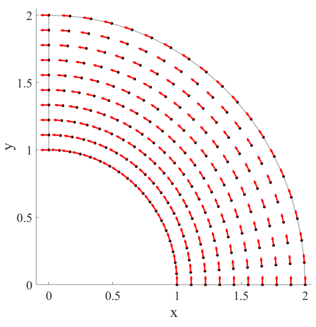
To further evaluate the effectiveness of the proposed data-driven computing framework in handling physical systems with a large variation in anisotropic orientations, an anisotropic cylinder subjected to internal pressure is analyzed, as shown in Fig. 5.9(a), which is composed of an orthotropic material with anisotropic orientations along the circumferential direction of the cylinder. Considering axisymmetry of the geometry and the applied load, only a quarter model shown in Fig. 5.9(b) is modeled. The inner and outer radius are 1 and 2, respectively. In Fig. 5.9(c), the anisotropic orientations of the 10×20 discretization points are denoted by red arrows . Two cases are examined and compared.

In Case 1, an isotropic elastic material is applied, with Young's modulus $E = 9 \times 10^3$ and Poisson's ratio $\nu = 0.2$. A noiseless synthetic isotropic material data set with 8000 strain-stress data points is generated by the phenomenological isotropic elastic material model with a range of [-0.4,0.4] for each strain component. In Case 2, an orthotropic elastic material is applied, with $E_1 = 4 \times 10^4, E_2 = 9 \times 10^3, \nu_{21} = 0.045, \nu_{12} = 0.2$, $G_{12} = 2 \times 10^4$, and anisotropic orientations along the circumferential direction of the cylinder. A noiseless synthetic orthotropic material data set with 8000 strain-stress data points is generated by the phenomenological orthotropic elastic material model with a range of [-0.4,0.4] for each strain component. Considering the uniform distribution of anisotropic orientations of material points in the cylinder, a rotated material database is constructed by rotating the generated synthetic data set from 90° to 180° with a 5° interval. Noisy material data sets are generated from the noiseless material data sets by the procedure described in Section 5.3.1. Internal pressure $p = 1000$ is applied and the data-driven analysis is performed with 20 loading steps.

The cross-sectional ($y = 0$) radial displacement (U_r) as well as the radial and the circumferential stress distributions of the data-driven solutions are compared with those of the constitutive model-based reference solutions, as shown in Figs. 5.10 - 5.12. The data-driven solutions obtained from using noisy material data sets are very close to those of reference



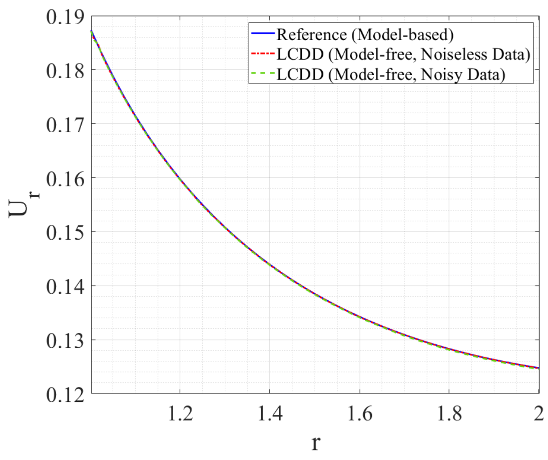
(a)



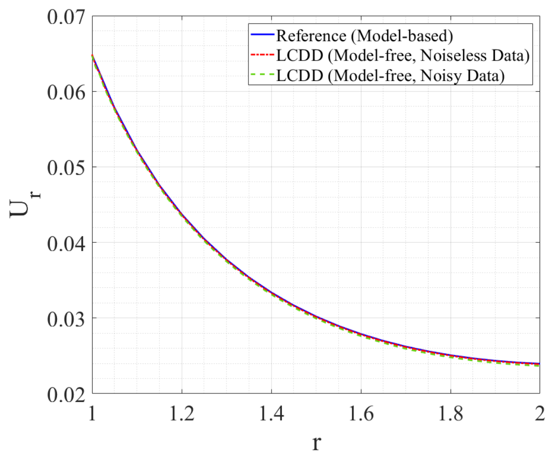
(b)

Figure 5.9. (a) Schematic of a cylinder subjected to internal pressure and a quarter model to be simulated; (b) red arrows denote nodal anisotropic orientations of material points in a discretization with 10×20 nodes

solutions, which again shows the robustness of the proposed data-driven framework to deal with noisy data sets. The results of both cases show a good agreement with the reference solutions, which demonstrates the effectiveness of the proposed data-driven framework in dealing with systems with a large variation in anisotropic orientations.

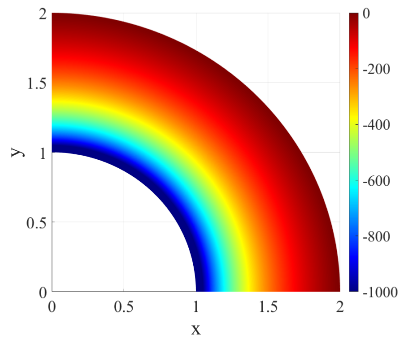


(a) Case 1

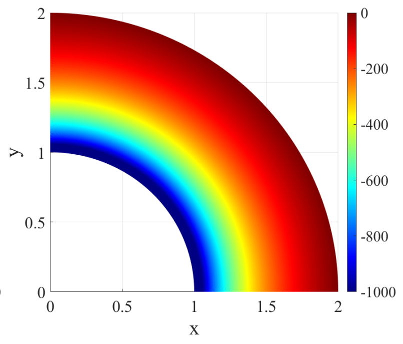


(b) Case 2

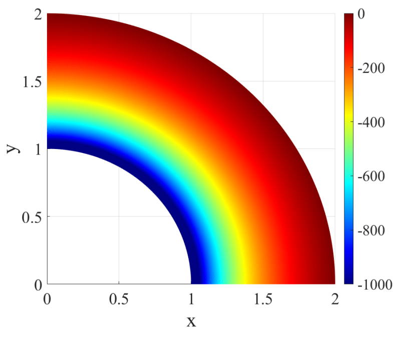
Figure 5.10. Comparison of data-driven solutions with constitutive model-based reference solutions. Cross-sectional radial displacement U_r: (a) Case 1; (b) Case 2



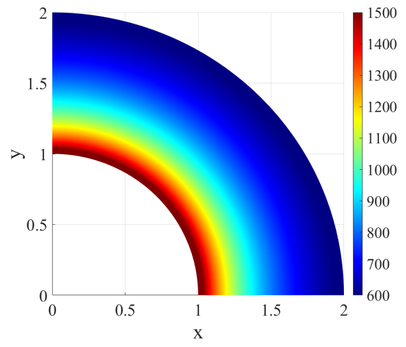
(a) Case 1 - Reference S_{rr}



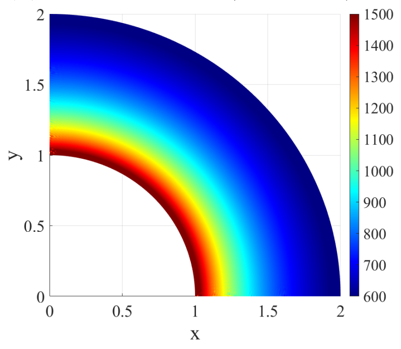
(b) Case 1 - LCDD (Noiseless) S_{rr}



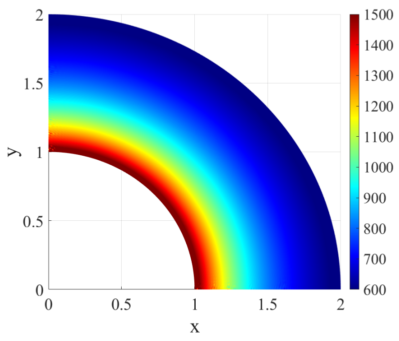
(c) Case 1 - LCDD (Noisy) S_{rr}



(d) Case 1 - Reference $S_{\theta\theta}$

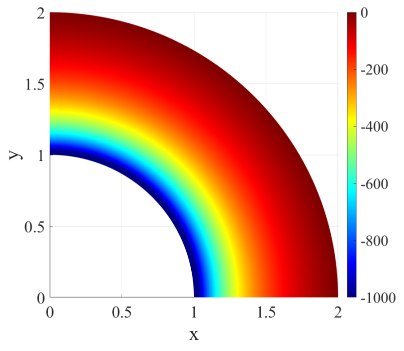


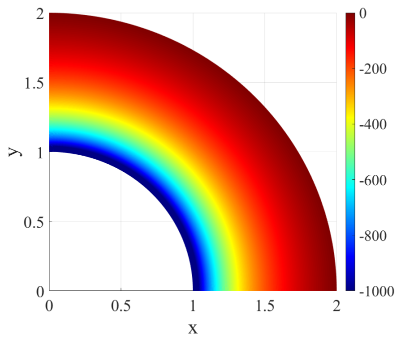
(e) Case 1 - LCDD (Noiseless) $S_{\theta\theta}$

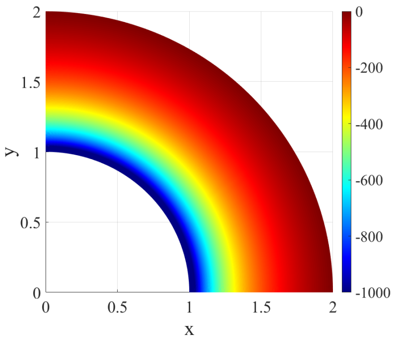


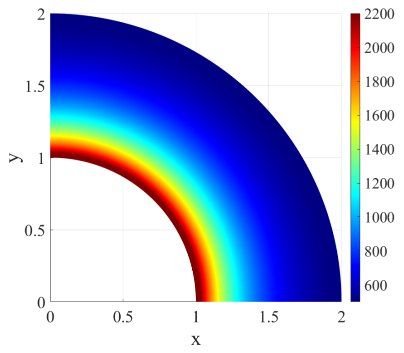
(f) Case 1 - LCDD (Noisy) $S_{\theta\theta}$

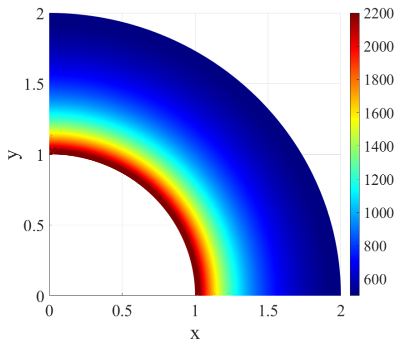
Figure 5.11. Comparison of data-driven solutions with constitutive model-based reference solutions of Case 1. Distribution of S_{rr} (stress in the radial direction) and $S_{\theta\theta}$ (stress in the circumferential direction): (a) Case 1: reference S_{rr}; (b) Case 1: data-driven solution with noiseless data of S_{rr}; (c) Case 1: data-driven solution with noisy data of S_{rr}; (d) Case 1: reference $S_{\theta\theta}$; (e) Case 1: data-driven solution with noiseless data of $S_{\theta\theta}$; (f) Case 1: data-driven solution with noisy data of $S_{\theta\theta}$


(a) Case 2 - Reference S_{rr}


(b) Case 2 - LCDD (Noiseless) S_{rr}


(c) Case 2 - LCDD (Noisy) S_{rr}


(d) Case 2 - Reference $S_{\theta\theta}$


(e) Case 2 - LCDD (Noiseless) $S_{\theta\theta}$

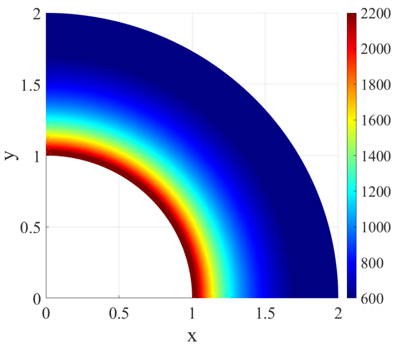

(f) Case 2 - LCDD (Noisy) $S_{\theta\theta}$

Figure 5.12. Comparison of data-driven solutions with constitutive model-based reference solutions of Case 1. Distribution of S_{rr} (stress in the radial direction) and $S_{\theta\theta}$ (stress in the circumferential direction): (a) Case 2: reference S_{rr}; (b) Case 2: data-driven solution with noiseless data of S_{rr}; (c) Case 2: data-driven solution with noisy data of S_{rr}; (d) Case 2: reference $S_{\theta\theta}$; (e) Case 2: data-driven solution with noiseless data of $S_{\theta\theta}$; (f) Case 2: data-driven solution with noisy data of $S_{\theta\theta}$

5.4 Summary

In this study, we develop a new data-driven material solver built upon the local convexity-preserving reconstruction scheme [38] to capture anisotropic material behaviors and enable data-driven modeling of anisotropic nonlinear elastic solids. The proposed data-driven approach assumes that the material data of anisotropic materials with a specific anisotropic orientation in a reference frame is accessible. The information of anisotropic orientations, e.g., the rotation angles between local fiber frames and the reference frame of the material data are utilized to construct an offline material database, which contains rotated material data sets representing anisotropic material properties with various anisotropic orientations. The offline rotated material database can be efficiently constructed and applied to data-driven simulations of anisotropic materials.

During online data-driven computing, a two-level local data search is integrated into the local convexity-preserving material solver. In the level-1 data search, two rotated material data sets with minimum anisotropic distance (Eq. (5.4)) to the material anisotropic orientation are identified as the local data sets. The selected rotated material data sets are considered to contain anisotropic material properties closest to that of the material point. In the level-2 data search, k data points closest to the given physical state based on strain-stress state distance (Eq. (4.15)) are obtained separately from the two rotated material data sets selected from the level-1 data search. A linear weighting scheme based on anisotropic distance is adopted to determine the number nearest data points from each of the selected material data sets. The final k nearest data points that contain information of strain, stress, and anisotropic orientations are used to construct a local convex space capturing the underlying anisotropic data structure, within which the optimal material data is reconstructed by the material solver. The optimally reconstructed material data is closest to the physical state of the material point in terms of both anisotropic distance and state distance and thus is considered to best represent the anisotropic properties of the material point.

The performance of the proposed data-driven computing framework is examined by two numerical examples, including deflection of a multi-layer anisotropic cantilever beam made of materials with different anisotropic orientations and inflation of an anisotropic cylinder where anisotropic orientations are along the circumferential direction of the cylinder. Synthetic noiseless and noisy material data are generated from the phenomenological material models and employed for the data-driven analysis for accuracy assessment of the proposed method. The data-driven solutions show a good agreement with the constitutive model-based reference solutions, which demonstrates its effectiveness and robustness of the proposed data-driven framework against noise present in the material data.

The proposed two-level data search can achieve high computational efficiency if the rotated material database is constructed offline such that online computation does not involve any frame transformation of states or data. For applications with a large amount of anisotropic material data and strong variations in anisotropic orientations, constructing a rotated material database with small anisotropic distance between rotated data sets requires a large amount of memory. In this case, more sophisticated metrics for anisotropic distance and reconstruction schemes will be investigated in order to achieve high accuracy in reconstructing anisotropic material properties from given material data sets that have a large anisotropic distance. In future studies, the proposed data-driven anisotropic modeling framework will be further examined with real-world data on three-dimensional material systems with aniostropic material behaviors, e.g., musculoskeletal systems consisting of muscle fibers with varying anisotropic orientations.

5.5 Acknowledgement

This chapter, in part, is a reprint of the material as it appears in: “Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, and Shantanu Sinha. Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids. *Data-Centric Engineering*, 1, 2020”. The dissertation author is the primary investigator and author of this paper.

Chapter 6

Deep autoencoders for physics-constrained data-driven nonlinear materials modeling

To counteract the curse of dimensionality, we propose to use autoencoders in the data-driven local solver for deep manifold learning of material data, allowing effective discovering of the underlying representation of stress-strain material data. To the best of the authors' knowledge, this is the first attempt to apply deep manifold learning in physics-constrained data-driven computing. In the following exposition, we demonstrate how autoencoder based deep learning enhances accuracy, robustness, and generalization ability of data-driven computing.

6.1 Introduction

In this work, we aim to develop a novel data-driven computing approach to overcome the curse of dimensionality and the lack of generalization in classical model-free data-driven computing approaches [2, 3, 28]. To this end, we propose to introduce a novel autoencoders based deep neural network architecture [44, 45] under the LCDD framework [3] to achieve two major objectives: dimensionality reduction and generalization of physically meaningful constitutive manifold. It should be emphasized that there have been various nonlinear dimensionality reduction (i.e. manifold learning) techniques developed for complex high-dimensional data [194–198], but these methods remain challenging in out-of-sample extension (OOSE), that is to project new unseen data onto the learned low-dimensional representation space, due to the lack of explicit mapping function. While nonparametric OOSE [198] is an option to construct the mapping, the computational complexity increases substantially with the size of dataset. On the other hand, owing to its deep learning architecture, an autoencoder is capable of naturally defining the mapping functions between high- and low-dimensional representation and capturing highly varying nonlinear manifold with good generalization capability [27, 198], as to be discussed in Section 6.2.

By integrating autoencoders and the discovered low-dimensional embedding into the data-driven solver, the proposed framework is referred to as auto-embedding data-driven (AEDD) computing, which can also be considered as a hybrid of the NN-based constitutive modeling and the classical model-free data-driven computing. In this approach, the autoencoders are first trained in an offline stage to extract a representative low-dimensional manifold (embedding) of the given material data. Autoencoders also provide effective noise filtering through the data compression processes. The trained autoencoders are then incorporated in the data-driven solver during the online computation with customized convexity-preserving reconstruction. Hence, all operations related to distance measure, including the search of the closest material points in the dataset are performed in the learned embedding space, circumventing the difficulties resulting from high dimensionality and data noise. To ensure numerical stability and representative constitutive manifold parameterized by the trained autoencoder networks, an efficient convexity-preserving interpolation is proposed to locally approximate the optimal material data to a given physically admissible state. Specifically, in this work, we present two different solvers to perform locally convex reconstruction, and demonstrate the one directly providing interpolation approximation without using decoders outperforms the one that fully uses the encoder-decoder network structure. Furthermore, it is shown that the proposed method is computationally tractable, since the additional autoencoder training is conducted offline and the online data-driven computation mainly involve lower-dimensional variables in the embedding space.

The remainder of this chapter is organized as follows. In section 6.2, the employment of autoencoders for deep material manifold learning is presented. Section 6.3 introduces the auto-embedding data-driven solvers and the efficient convexity-preserving interpolation for improved performance. Finally, in Section 6.4, the effectiveness of the proposed AEDD framework are examined, and a parametric study is conducted to investigate the effects of autoencoder architecture, data noise, data size and sparsity, and neural network initialization on AEDD's performance. The proposed method is also applied to biological tissue modeling to demonstrate the enhanced effectiveness and generalization capability of AEDD over the other data-driven schemes. Concluding remarks and discussions are summarized in Section 6.5.

6.2 Autoencoders for low-dimensional manifold of material data

The basic concepts of deep neural networks (DNNs) and autoencoders have been introduced in Section 2.1.1 and Section 2.1.2, respectively. For effective data search in the local step, Eq. (4.3) or Eqs. (4.14) and (4.16), we present the employment of autoencoders to construct low-dimensional nonlinear representation (embedding) of material data. Thereafter, optimum data search on the low-dimensional data manifold using a locally convex projection method is presented in Section 6.3.

6.2.1 Nonlinear material embedding

In this study, autoencoders are used to discover the intrinsic low-dimensional material embedding of the given material dataset $\mathbb{E} = \{\hat{\mathbf{z}}_I\}_{I=1}^{M}$, where $\hat{\mathbf{z}}_I = (\hat{\mathbf{E}}_I, \hat{\mathbf{S}}_I)$ and M is the number of material data points. Given the autoencoder architecture $\mathbf{h}(\cdot;\theta_{\text{enc}},\theta_{\text{dec}})$ in Eq. (2.6), the parameters θ^*_{enc} and θ^*_{dec} are computed by minimizing the loss function in Eq. (2.7).

The training procedures of autoencoders in terms of the loss function in Eq. (2.7) are performed offline. Thus, training on a large material dataset does not result in additional overhead on the online data-driven computation. The details of the training algorithms for autoencoders are given in Section 6.2.2.

Remark. *It is well known that for an autoencoder with a single hidden layer and linear activation* *function, the weights trained by the mean-squared-error cost function learn to span the same principal subspace as principal components analysis (PCA) [199]. Autoencoders based on neural networks with nonlinear transform functions can be thought of as a generalizaiton of PCA, capable of learning nonlinear relationships.*

Given the trained autoencoder $\mathbf{h}(\cdot;\theta^*_{\text{enc}},\theta^*_{\text{dec}})$, we can define a low-dimensional embedding space, $\mathscr{E}' = \{\mathbf{z}' \in \mathbb{R}^p \,|\, \mathbf{z}' = \mathbf{h}_{\text{enc}}(\mathbf{z};\theta^*_{\text{enc}}), \forall \mathbf{z} \in \mathscr{Z}\}$, in which the material state is described by a lower-dimensional coordinate system $\mathbf{z}'$. Here, the prime symbol $(\cdot)'$ is used to denote the quantities defined in the embedding space, and $\mathscr{Z}$ denotes the high-dimensional phase space where the material states $\hat{\mathbf{z}}$ and the physical states $\mathbf{z}$ are defined. For example, the embedding set of the given material data is

$$\mathbb{E}' = \{\hat{\mathbf{z}}'_I\}_{I=1}^{M} \subset \mathscr{E}', \tag{6.1}$$

where $\hat{\mathbf{z}}'_I = \mathbf{h}_{\text{enc}}(\hat{\mathbf{z}}_I;\theta^*_{\text{enc}})$ for $\hat{\mathbf{z}}_I \in \mathbb{E}$.

Considering the data-driven application on learning the underlying structure of material data, autoencoders provide the following advantages:

1) Deep neural network architecture enables autoencoders to capture highly complex nonlinear manifold with exponentially less data points than nonparametric methods based on nearest neighbor graph [27, 198, 200].

2) Autoencoders provide explicit mapping functions, i.e. $\mathbf{h}_{\text{enc}}$ and $\mathbf{h}_{\text{dec}}$, between the high- and low-dimensional representation so that the trained encoders allow efficient evaluation of the embedding of new input data.

3) Through information compression by encoders, unwanted information of material data, such as noise and outliers, can be filtered while preserving its essential low-dimensional manifold structure [27].

Compared to data-driven methods based on conventional manifold learning techniques [3, 28, 201], the explicit nonlinear mapping functions learned by autoencoders are particularly attractive to data-driven computing because not only can they encode the essential global structure of the given material data for enhanced generalization ability, they also greatly reduce online computational cost by using the pretrained autoencoders. Furthermore, as we can see in next section, due to the availability of low-dimensional embedding $\mathscr{E}'$, we can introduce a convexity-preserving interpolation scheme to effectively search for the optimal material data associated with the given physical state.

6.2.2 Autoencoder architectures and training algorithms

As the architectures of encoder and decoder in an autoencoder are symmetric, we only use the encoder architecture to denote the autoencoder architecture. For example, the encoder architecture in Fig. 2.4 is $4-6-4-3$, where the first and last values denote the numbers of artificial neurons in the input layer and embedding layer, respectively, and the other values denote the neuron numbers of the hidden layers in sequence. As such, the decoder architecture in this case is $3-4-6-4$.

The offline training on the given material datasets is performed by using the open-source Pytorch library [202], and the optimal parameters θ^*_{enc} and θ^*_{dec} of autoencoders are obtained by minimizing the loss function (Eq. (2.7)). The regularization parameter β is set as 10^{-5}. A hyperbolic tangent function is adopted as the activation function for all layers of autoencoders, except for the embedding layer and the output layer, where a linear function is employed instead. To eliminate the need of manually tuning the learning rate for training, an adaptive gradient algorithm, Adagrad [203], is employed, where the initial learning rate is set to be 0.1 and the number of training epochs is set to be 2000. The training datasets are standardized such that they have zero mean and unit variance to accelerate the training process. It should be noted that the training of autoencoders could get trapped in local minima and this can be overcome by pretraining the network using Restricted Boltzmann Machines or by denoising autoencoders [198, 204, 205].

6.3 Auto-embedding data-driven (AEDD) solver

We now develop the AEDD solver based on autoencoders to search for the optimal material data in the solution process of the local step (e.g., Eqs. (4.14) or (4.16)). We begin with introducing a simple interpolation scheme to preserve local convexity in the material data search, which is essential in enhancing the local solver performance, followed by presenting two AEDD solvers with the employment of convexity-preserving reconstruction.

6.3.1 Convexity-preserving interpolation

With deep manifold learning by autoencoders, we are able to extract the underlying low-dimensional global manifold of material datasets, and to enhance the generalization capability of the material local solver. During data-driven computing, material and physical states are projected onto the constructed material embedding space $\mathscr{E}'$, and a convexity-preserving local data reconstruction is introduced for enhanced stability and convergence in the local data search on the embedding space.

In this approach, because the material embedding points in low-dimensional space are explicitly given by the offline trained autoencoders, interpolation schemes on the embedding space is straightforward without suffering the high-dimensionality issues. A convexity-preserving, partition-of-unity interpolation method is therefore introduced into the material data-driven solver, using Shepard function [206] or inverse distance weighting. Shepard interpolation has been widely used in data fitting and function approximation with positivity constraint [207–209].

Here, the Shepard functions are applied to reconstruct the material embedding of a given physical state, $\mathbf{z}' = \mathbf{h}_{\text{enc}}(\mathbf{z}; \theta^*_{\text{enc}})$, by its material embedding neighbors, expressed as

$$\mathbf{z}'_{recon} = \mathscr{I}\left(\{\Psi_I(\mathbf{z}'); \hat{\mathbf{z}}'_I\}_{I \in \mathscr{N}_k(\mathbf{z}')}\right) = \sum_{I \in \mathscr{N}_k(\mathbf{z}')} \Psi_I(\mathbf{z}')\hat{\mathbf{z}}'_I, \tag{6.2}$$

where $\mathbf{z}'_{recon}$ is the reconstruction of $\mathbf{z}'$, $\hat{\mathbf{z}}'_I$ is the material data embedding in $\mathbb{E}'$ defined in Eq.

(6.1), $\mathscr{N}_k(\mathbf{z}')$ is the index set of the k nearest neighbor points of $\mathbf{z}'$ selected from $\mathbb{E}'$, and the shape functions are

$$\Psi_I(\mathbf{z}') = \frac{\phi(\mathbf{z}' - \hat{\mathbf{z}}'_I)}{\sum_{J=1}\phi(\mathbf{z}' - \hat{\mathbf{z}}'_J)}. \tag{6.3}$$

In Eqs. (6.2) and (6.3), ϕ is a positive kernel function representing the weight on the data set $\{\hat{\mathbf{z}}'_I\}_{I\in\mathscr{N}_k(\mathbf{z}')}$, and $\mathscr{I}$ denotes the interpolation operator that constructs shape functions with respect to $\mathbf{z}'$ and its neighbors. Note that these functions form a partition of unity, i.e., $\sum_{I\in\mathscr{N}_k(\mathbf{z}')}\Psi_I(\mathbf{z}') = 1$ for transformation objectivity. Furthermore, they are convexity-preserving when the kernel function ϕ is a positive function. Here, an inverse distance function is used as the kernel function

$$\phi(\mathbf{z}' - \hat{\mathbf{z}}'_I) = \frac{1}{||\mathbf{z}' - \hat{\mathbf{z}}'_I||^2}. \tag{6.4}$$

It is worth noting that the interpolation functions defined in Eqs. (6.3) and (6.4) are equivalent to the RK approximation function in (4.25) with zero-order basis.

Fig. 6.1 demonstrates the locally convex reconstruction by the proposed interpolation in Eq. (6.2). For example, the given blue asterisk is mapped to the blue-square point by using the Shepard interpolation. It can be seen that the three given points (inside (red), on-edge (pink), and outside (blue)) are all mapped to locations within the convex hull, showing the desired convexity-preserving capability. The interpolation is simple and efficient as the interpolation functions in Eq. (6.3) can be constructed easily in a low-dimensional embedding space.

6.3.2 Auto-embedding data-driven (AEDD) solver in data-driven computing

The physics-constrained data-driven computing described in Chapter 4 is conducted in the high-dimensional phase space $\mathscr{Z}$ (or called *data space*), where the physical state $\mathbf{z}_\alpha \in \mathscr{C}$, the material data $\hat{\mathbf{z}}_\alpha \in \mathscr{E}$ and the material dataset $\mathbb{E}$ are defined in $\mathscr{Z}$. We use the subscript "α" to denote the quantities at integration points with the employment of numerical discretization, see Section 4.2. To enhance solution accuracy and generalization capability of data-driven

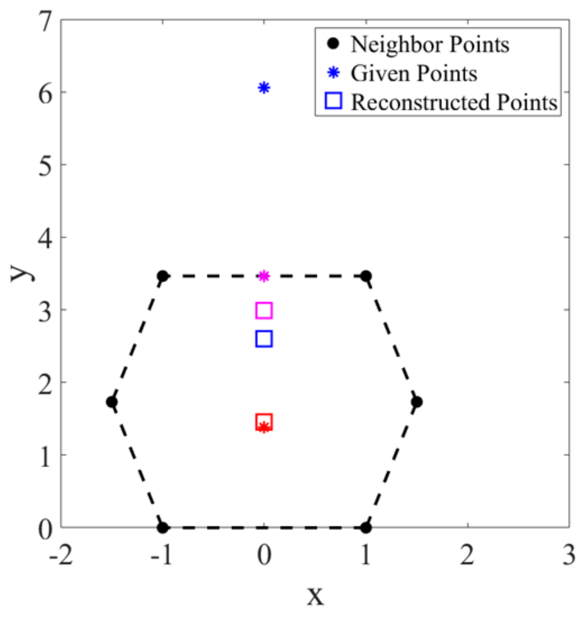


Figure 6.1. Demonstration of the convexity-preserving reconstruction by Shepard interpolation in Eq. (6.2), where the asterisk and square denote the given and the reconstructed points, respectively, black dots represent the nearest neighbors of the given points, and the black dash line depicts a locally convex hull formed by the nearest neighbors.

computing, deep manifold learning enabled by autoencoders is introduced into the material data-driven local solver.

Recall that autoencoders introduced in Section 6.2.1 are trained offline and the trained encoder $\mathbf{h}_{\text{enc}}$ and decoder $\mathbf{h}_{\text{dec}}$ functions are employed directly in the online data-driven computation. As such, the encoder maps an arbitrary point from the data space to the embedding space, i.e. $\mathbf{z}'_\alpha = \mathbf{h}_{\text{enc}}(\mathbf{z}_\alpha)$, whereas the decoder performs the reverse mapping, i.e. $\tilde{\mathbf{z}}_\alpha = \mathbf{h}_{\text{dec}}(\mathbf{z}'_\alpha)$. With the autoencoders and the proposed convexity-preserving data reconstruction in the embedding space introduced in Section 6.3.1, we propose the following two AEDD approaches for the material data-driven local solver. The objective is to find the optimal material data for a given physical state $\mathbf{z}_\alpha$ computed in Section 4.2.

AEDD local solver: Solver I

Let $\mathbf{z}_\alpha$ be the physical state obtained from the global step with physical constraints, Eq. (4.7), or the corresponding variational equations, Eqs. (4.12)–(4.13). We first introduce a local solver that uses decoders for reverse mapping from the embedding space to the data space, denoted as Solver I. In this approach, the local problem defined in Eq. (4.3) is reformulated by three steps, as described below:

$$\textit{Step 1}: \qquad \mathbf{z}'_\alpha = \mathbf{h}_{\text{enc}}(\mathbf{z}_\alpha), \tag{6.5a}$$

$$\textit{Step 2}: \qquad \hat{\mathbf{z}}'^*_\alpha = \mathscr{I}\left(\{\Psi_I(\mathbf{z}'_\alpha);\hat{\mathbf{z}}'_I\}_{I\in\mathscr{N}_k(\mathbf{z}'_\alpha)}\right) \tag{6.5b}$$

$$\textit{Step 3}: \qquad \hat{\mathbf{z}}^*_\alpha = \mathbf{h}_{\text{dec}}(\hat{\mathbf{z}}'^*_\alpha), \tag{6.5c}$$

for $\alpha = 1,...,N$, where $\hat{\mathbf{z}}'_I \in \mathbb{E}'$ (see Eq. (6.1)), and $\mathscr{I}$ is the convexity-preserving interpolation operator defined in Eq. (6.2).

The schematic of data-driven computing with Solver I is illustrated in Fig. 6.2(a), where the integration point index α is dropped for brevity. For example, at the v-th global-local iteration, after the physical state $\mathbf{z}^{(v)}$ (the blue-filled triangle) is obtained from the global physical step (Eq. (4.7)), *Step 1* of the local solver (Eq. (6.5a)) maps the sought physical state from the data space to the embedding space by the encoder, $\mathbf{z}'^{(v)} = \mathbf{h}_{\text{enc}}(\mathbf{z}^{(v)})$, depicted by the white-filled triangle in Fig. 6.2(a). In *Step 2*, k nearest neighbors of $\mathbf{z}'^{(v)}$ based on Euclidean distance are sought in the embedding space and the optimal material embedding solution $\hat{\mathbf{z}}'^{*(v)}$ (the red square) is reconstructed by using the proposed convexity-preserving interpolation (Eqs. (6.2)-(6.4)). Lastly, in *Step 3*, the optimal material embedding state $\hat{\mathbf{z}}'^{*(v)}$ is transformed from the embedding space to the data space by the decoder, $\hat{\mathbf{z}}^{*(v)} = \mathbf{h}_{\text{dec}}(\hat{\mathbf{z}}'^{*(v)})$ (the red star in Fig. 6.2(a)). Subsequently, this resultant material data solution $\hat{\mathbf{z}}^{*(v)}$ from the local solver in Eq. (6.5) is used in the next physical solution update $\mathbf{z}^{(v+1)}$. These processes complete one global-local iteration. The iterations proceed until the distance between the physical and material states is within a tolerance, yielding the data-driven solution denoted by the green star in Fig. 6.2(a), which ideally is the intersection between the physical manifold and material manifold in the data space.

Here, the nearest neighbors searching and locally convex reconstruction of the optimal material state are processed in the filtered (noiseless) low-dimensional embedding space, resulting in the enhanced robustness against noise and accuracy of the local data-driven solution.

AEDD local solver: Solver II

Although autoencoders aim to transform the input material data to output data with maximally preserving essential features, see Fig. 2.4, the decoder functions $\mathbf{h}_{\text{dec}}$ do not exactly reproduce the given material data in the data space due to the information compression and errors inevitably introduced by training processes [27]. During *Step 3* of Solver I (Eq. (6.5c)), the material embedding solution $\hat{\mathbf{z}}'^{*}_{\alpha}$ in Eq. (6.5b) projecting back to the data space by decoders could involve data reconstruction errors. That is, the performance of AEDD Solver I is subject to the quality of the trained decoder functions.

To enhance the robustness and stability of data-driven computing, we propose the second AEDD local solver (Solver II) that circumvents the use of decoders and, instead, uses the interpolation scheme in Eq. (6.2) to perform locally convex reconstruction directly on material dataset. The procedures of this solver are expressed as

$$\textit{Step 1}: \qquad \mathbf{z}'_{\alpha} = \mathbf{h}_{\text{enc}}(\mathbf{z}_{\alpha}), \tag{6.6a}$$

$$\textit{Step 2}: \qquad \hat{\mathbf{z}}^{*}_{\alpha} = \mathscr{I}\left(\{\Psi_I(\mathbf{z}'_{\alpha}); \hat{\mathbf{z}}_I\}_{I \in \mathscr{N}_k(\mathbf{z}'_{\alpha})}\right), \tag{6.6b}$$

for $\alpha = 1, ..., N$, where $\hat{\mathbf{z}}_I \in \mathbb{E}$ are the material data given in the original data space. The key ingredient of this approach is that the modified locally convex reconstruction in *Step 2* involves interpolation functions constructed in embedding space but interpolating material data that are in data space. It can be viewed as a blending interpolation approach compared to that in Solve I. The effectiveness of Solver I and II will be compared and discussed in Section 6.4.2.

In both Solver I and II, the interpolation functions $\Psi_I(\mathbf{z}'_{\alpha})$ are evaluated on the embedding space related to the embedded physical state $\mathbf{z}'_{\alpha}$ and its k nearest neighbors in the material embedding data $\{\hat{\mathbf{z}}'_I\}_{I \in \mathscr{N}_k(\mathbf{z}'_{\alpha})} \subset \mathbb{E}'$. In Solver II, however, these functions are weighted on the un-projected material data $\{\hat{\mathbf{z}}_I\}_{I \in \mathscr{N}_k(\mathbf{z}'_{\alpha})} \subset \mathbb{E}$ corresponding to the k selected neighbors. Because the

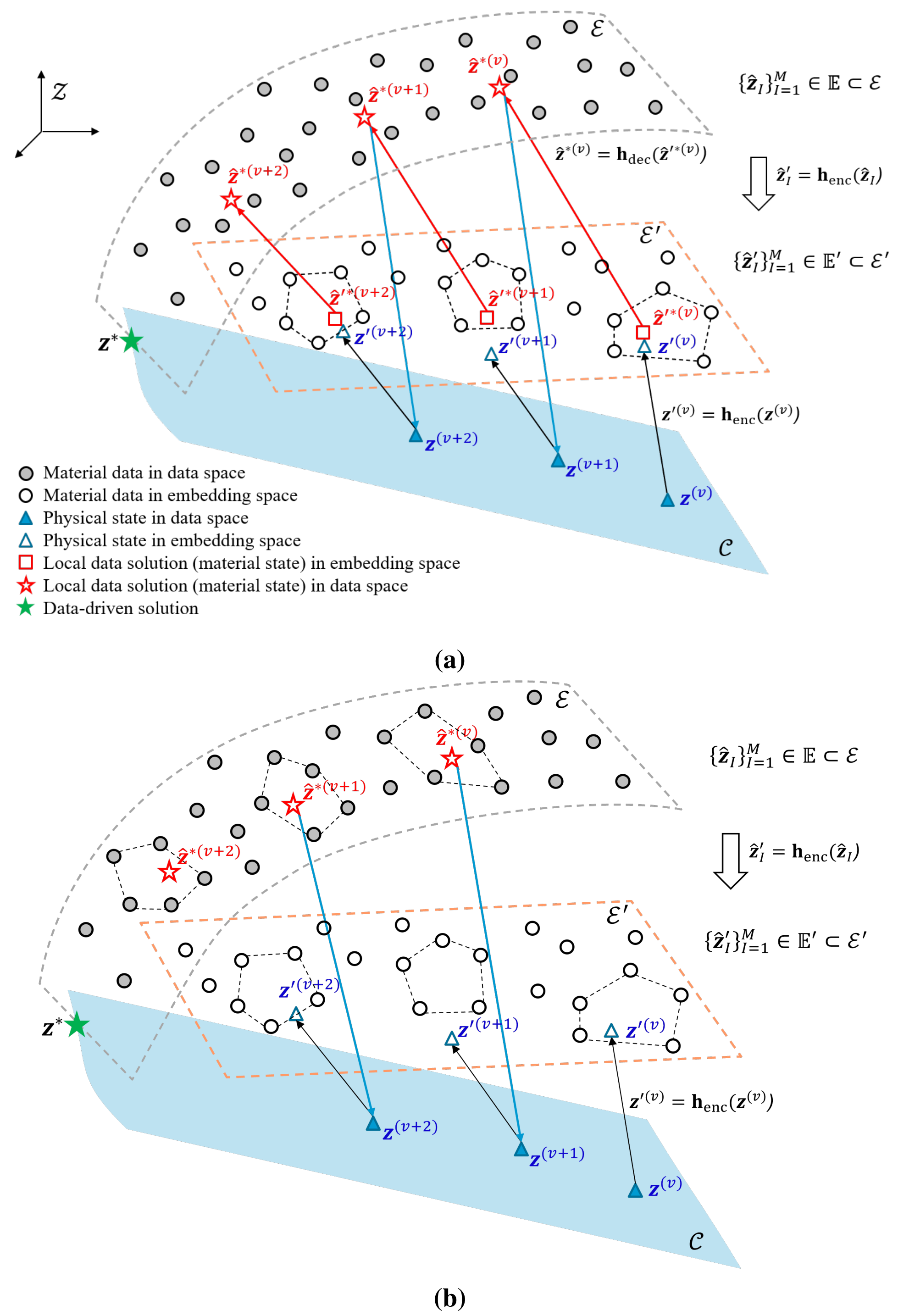

$\mathcal{Z}$
$\mathcal{E}$
$\{\hat{\mathbf{z}}_I\}_{I=1}^M \in \mathbb{E} \subset \mathcal{E}$
$\hat{\mathbf{z}}^{*(v)} = \mathbf{h}_{\text{dec}}(\hat{\mathbf{z}}'^{*(v)})$
$\hat{\mathbf{z}}'_I = \mathbf{h}_{\text{enc}}(\hat{\mathbf{z}}_I)$
$\mathcal{E}'$
$\{\hat{\mathbf{z}}'_I\}_{I=1}^M \in \mathbb{E}' \subset \mathcal{E}'$
$\hat{\mathbf{z}}^{*(v)}$
$\hat{\mathbf{z}}^{*(v+1)}$
$\hat{\mathbf{z}}^{*(v+2)}$
$\hat{\mathbf{z}}'^{*(v)}$
$\hat{\mathbf{z}}'^{*(v+1)}$
$\hat{\mathbf{z}}'^{*(v+2)}$
$\mathbf{z}'^{(v)}$
$\mathbf{z}'^{(v+1)}$
$\mathbf{z}'^{(v+2)}$
$\mathbf{z}'^{(v)} = \mathbf{h}_{\text{enc}}(\mathbf{z}^{(v)})$
$\mathbf{z}^{(v)}$
$\mathbf{z}^{(v+1)}$
$\mathbf{z}^{(v+2)}$
$\mathbf{z}^*$
$\mathcal{C}$
Material data in data space
Material data in embedding space
Physical state in data space
Physical state in embedding space
Local data solution (material state) in embedding space
Local data solution (material state) in data space
Data-driven solution
(a)
(b)

Figure 6.2. Geometric schematic of the proposed auto-embedding data-driven computational framework: (a) Solver I; (b) Solver II, corresponding to two different ways to reconstruct the optimal material data solution in high-dimensional data space. The material data points (the gray-filled circles), $\hat{\mathbf{z}}_I$, in the phase space are related to the material embedding points (the white-filled circles) $\hat{\mathbf{z}}'_I$ via the encoder function. The low-dimensional embedding manifold is represented by the orange dash line.

locally convex reconstruction in Eq. (6.6b) gives the optimal material data solution immediately in the data space, the decoder is avoided in Solver II.

Fig. 6.2(b) shows a schematic of the proposed data-driven computing based on Solver II. Taking the v-th global-local iteration as an example, the physical state $\mathbf{z}^{(v)}$ obtained from the global physical step (Eq. (4.7)) is mapped to the embedding space $\mathbf{z}'^{(v)}$ by the encoder. In *Step 2* of Solver II, the same k nearest neighbors search is performed on the embedding space $\mathscr{E}'$, while their corresponding material data in the original dataset are used in the data reconstruction via Eqs. (6.2)–(6.4). As shown in Fig. 6.2(b), the locally convex reconstruction of the material embedding state $\mathbf{z}'^{(v)}$ can be directly performed with the material data $\{\hat{\mathbf{z}}_I\}_{I\in\mathscr{N}_k(\mathbf{z}'_\alpha)}$ in the data space by using data indices, yielding the optimal material solution $\hat{\mathbf{z}}^{*(v)}$.

It is worth emphasizing that in comparison with the LCDD approach [3], the key difference in the proposed solver is that the neighbor search and data reconstruction are performed on the embedding space $\mathscr{E}'$, a lower-dimensional space constructed by the pre-trained encoder function. Thus, the proposed AEDD with Solver II can be considered as a enhanced generalization of LCDD for high-dimensional material data. We use this approach as the default AEDD method, unless stated otherwise.

6.4 Numerical results

In this section, the proposed AEDD approach is first tested on a cantilever beam using synthetic material data generated by constitutive laws. In this example, the effects of several factors on autoencdoers and the resulting AEDD data-driven solutions are investigated, including the size, sparsity, and the noise level of material datasets, neural network initialization during autoencoder training, and autoencoder architectures, aiming to validate the robustness and reliability. In the second subsection, AEDD is applied to modeling biological tissues using experimental data measured from heart valve tissues to demonstrate the enhanced generalization capability.

For simplicity, we consider homogeneous material in the following numerical examples, and thus the same material dataset, e.g., $\mathbb{E} = \{\hat{\mathbf{z}}_I\}_{I=1}^M = \{(\hat{\mathbf{E}}_I, \hat{\mathbf{S}}_I)\}_{I=1}^M$, with M data points, is used for all integration points.

6.4.1 Cantilever beam: Verification of the AEDD method

To verify the proposed AEDD framework (with Solver II in Section 6.3.2 by default), a cantilever beam subjected to a tip shear load is analyzed, as shown in Fig. 6.3. The Saint Venant-Kirchhoff phenomenological model with Young's modulus $E = 4.8 \times 10^3 N/mm^2$ and Poisson's ratio $\nu = 0$ is used to generate material datasets for training autoencoders. The problem domain is discretized with 41×5 randomly distributed nodes. The data-driven analysis is performed with 10 equal loading steps under a plane-strain condition. Following the same setting in [3], the weight matrix $\hat{\mathbb{C}}$ used in the distance metric (Eqs. (4.4)-(4.5)) and the physical solver (Eq. (4.13)-(4.12)) is defined as

$$\hat{\mathbb{C}} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \tag{6.7}$$

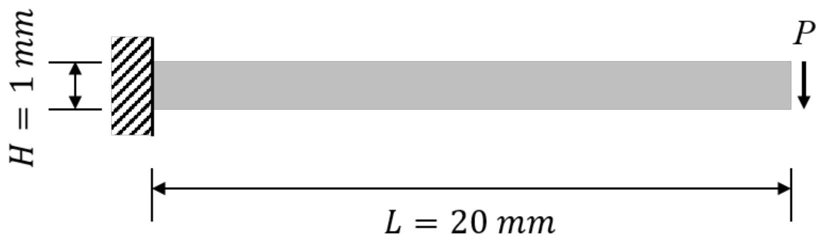


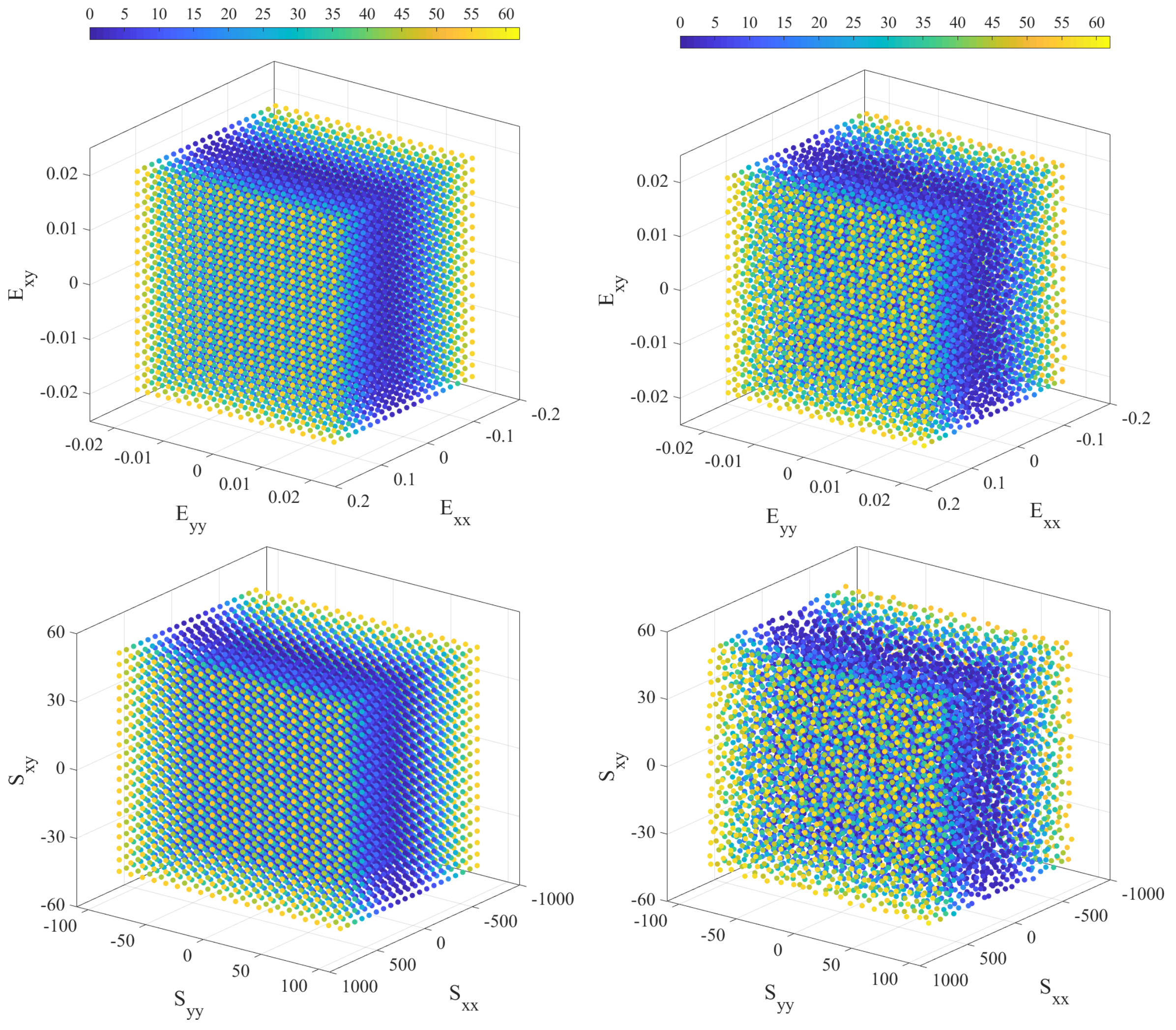
Figure 6.3. Schematic of a cantilever beam model subjected to a tip shear load, where $P = 10EI/L^2$, and $I = H^3/12$.

Preparation of material datasets

To assess the robustness and convergence property of AEDD against noise presented in the given material datasets, four manufactured noisy material datasets approximating the Saint Venant-Kirchhoff phenomenological model with different data sizes, i.e. $M = 10^3$, 20^3, 30^3, and 40^3, are considered. The generation procedure of these noisy datasets is described below. First, an noiseless dataset, $\bar{\mathbb{E}} = \{\bar{\mathbf{z}}_I\}_{I=1}^{M}$ is generated, where each Green-Lagrangian strain component is uniformly distributed within the range $[-0.02, 0.02]$ and the 2nd-PK stress components are obtained by using the elastic tensor in Eq. (6.7) that relates strain to stress. The example with $M = 20^3$ is shown in Fig. 6.4(a), where the strain and stress components are displayed separately for visualization. Following [3, 103], Gaussian perturbations scaled by a factor dependent on the size of datasets, $0.4\bar{\mathbf{z}}_{max}/\sqrt[3]{M}$, are added to each component of the noiseless dataset $\bar{\mathbb{E}}$ to obtain the associated noisy datasets $\mathbb{E} = \{\hat{\mathbf{z}}_I\}_{I=1}^{M}$, where $\bar{\mathbf{z}}_{max}$ is a vector of the maximum values for each component among the noiseless dataset. The noisy dataset corresponding to $M = 20^3$ is shown in Fig. 6.4(b). Fig. 6.5 shows the other three noisy material datasets.

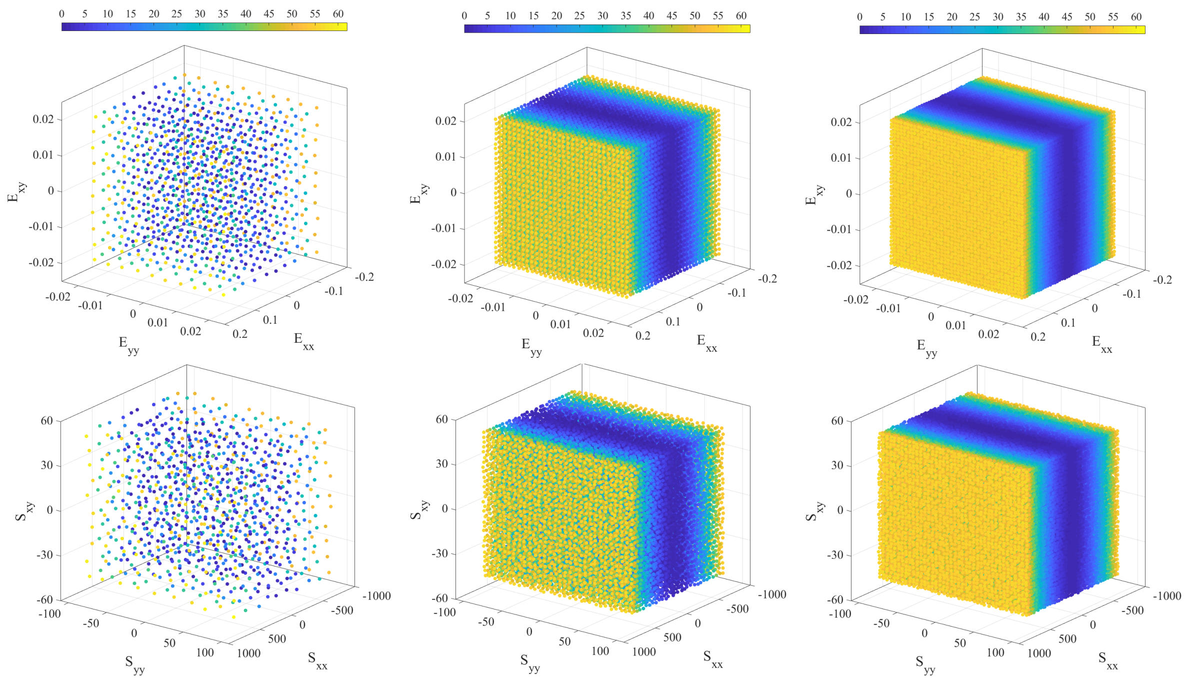
Effects of autoencoder architecture and initialization

In order to assess the effects of initialization during training and network architectures on autoencoders' accuracy and robustness, five random initializations and four architectures of autoencoders are considered. For the given noisy material datasets associated with this plane-strain cantilever beam problem, it is observed that autoencoders with an embedding layer of the dimension $p = 1$ or $p = 2$ could not capture a meaningful embedding representation. This is consistent to the observation in [3] where the number of neighbor points to construct the locally convex embedding is suggested to be larger than the number of intrinsic dimensionality, which is 2 of the employed linear elastic database. Hence, it requires the embedding dimension to be greater than 2. As described in Section 6.2.2, the encoder architecture is used to represent the autoencoder architecture. Four encoder architectures, $6-4-3$, $6-5-4$, $6-5-4-3$, and $6-10-8-5$, are considered in the following tests.



(a) Noiseless **(b)** Noisy

Figure 6.4. Material dataset with a size of $M = 20^3$: (a) Noiseless; (b) Noisy; Top: strain components; Bottom: stress components

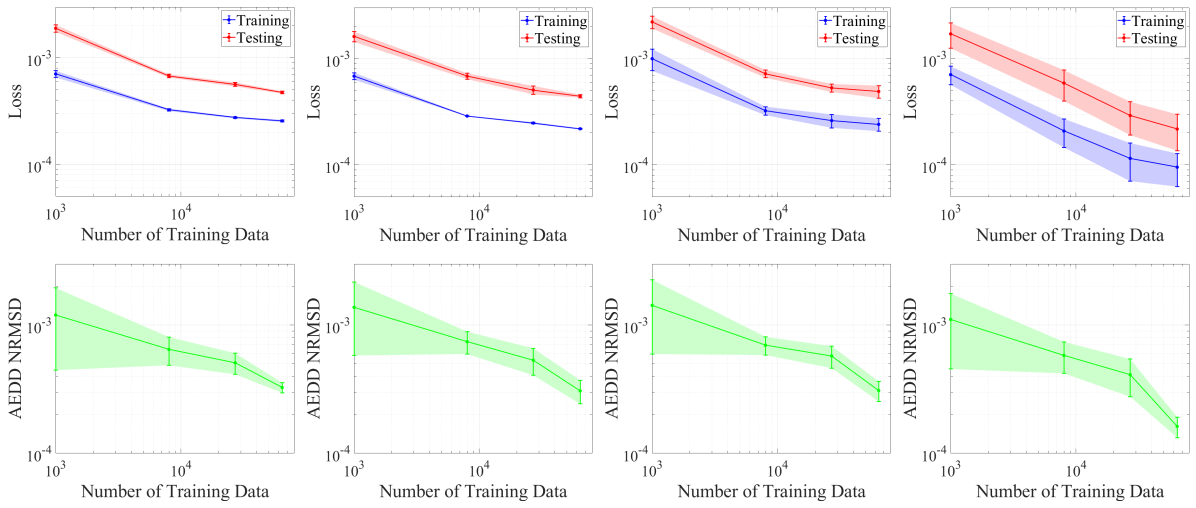


(a) **(b)** **(c)**

Figure 6.5. Noisy material datasets with: (a) $M = 10^3$; (b) $M = 30^3$; (c) $M = 40^3$; Top: strain components; Bottom: stress components

The first row in Fig. 6.6 shows the error curves (mean with standard deviations shaded) of the final training and testing losses against the size of training dataset for different autoencoder architectures. Here, the noisy material datasets of different sizes, $M = 10^3$, 20^3, 30^3, and 40^3, that defined in Section 6.4.1 are used for training the autoencoders, where autoencoders are trained with five random initialization for each case. Besides, to fairly compare the testing errors between the autoencoders trained with various sizes of training data, we use the same test dataset consisting of 729 material data points that are generated from the same procedure in Section 6.4.1 but not included in the given material datasets.

As we can see, all the selected autoencoders converge well, yielding smaller training and testing errors as the size of material dataset increases. Moreover, it is observed that the autoencoder with a larger architecture could lead to greater variation due to training randomness, indicated by the standard deviations. This is because the training algorithms used to minimize the loss function in Eq. (2.7) do not guarantee global minimization, and a larger DNN with more trainable parameters may cause higher randomness. However, the trained results shown here are satisfactory due to the employment of regularization. It also shows that the training and testing losses decrease as the dimension of the embedding layer increases.



(a) Encoder: 6-4-3 **(b)** Encoder: 6-5-4 **(c)** Encoder: 6-5-4-3 **(d)** Encoder: 6-10-8-5

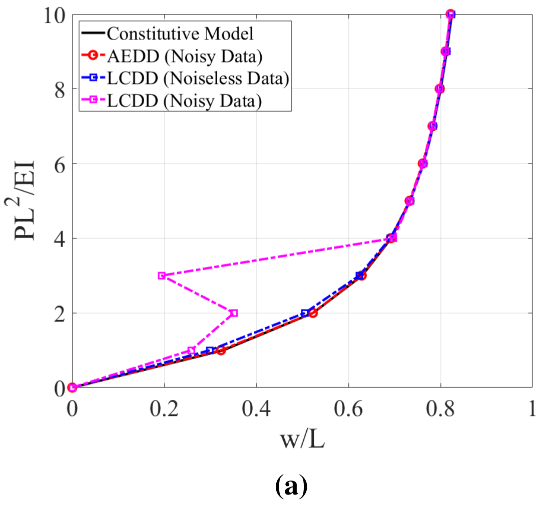
Figure 6.6. Error curves (mean with standard deviations shaded) of four different encoder architectures: (a) $6-4-3$; (b) $6-5-4$; (c) $6-5-4-3$; (d) $6-10-8-5$ Top: final training and testing losses of autoencoders; Bottom: *NRMSD* between AEDD and constitutive model-based solutions

Data-driven modeling results

The data-driven solution is compared with the constitutive model-based reference solution using Eq. (6.7). To better assess the accuracy of AEDD solutions, a normalized root-mean-square deviation (*NRMSD*) is introduced

$$NRMSD = \sqrt{\sum_{i}^{N_{eval}} \frac{(\bar{w}_i^{AEDD} - \bar{w}_i^{ref})^2}{N_{eval}}} / (PL^2/EI), \tag{6.8}$$

where $N_{eval} = 200$ is the number of evaluation points, $\bar{w}_i^{AEDD}$ and $\bar{w}_i^{ref}$ are the normalized tip deflection obtained by AEDD and model-based reference solutions, respectively. In this cantilever beam case, the normalized tip deflection $\bar{w}_i = w_i/L$ is obtained at the maxiumn loading, i.e. $PL^2/EI = 10$, where L and H are the length and the width of the beam, respectively, and $I = H^3/12$, see Fig. 6.3.



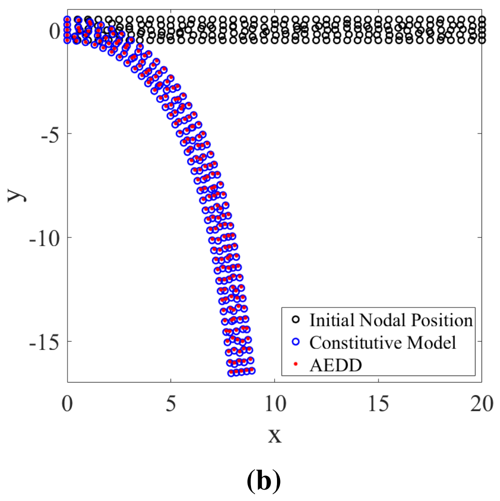
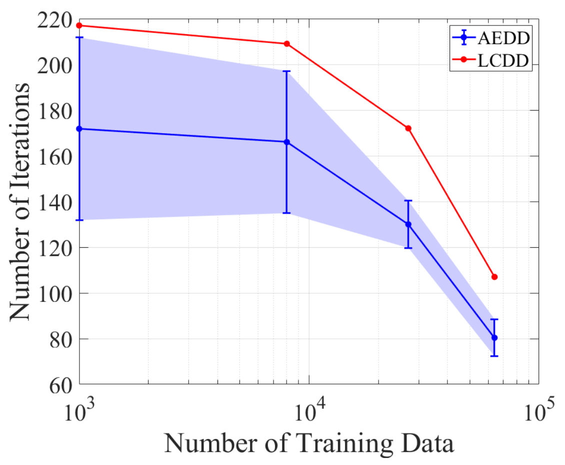
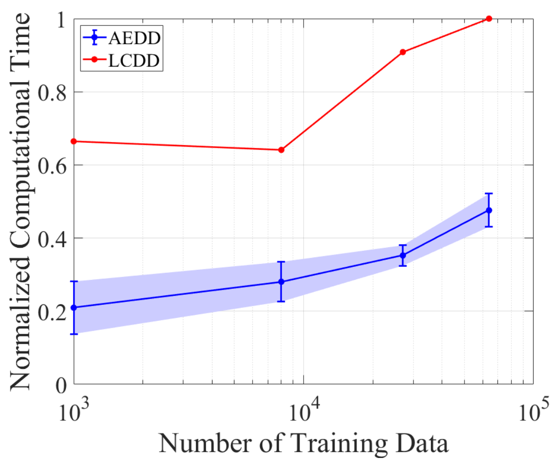


Figure 6.7. Comparison of constitutive model-based, LCDD, and AEDD solutions: (a) normalized tip deflection-loading; (b) initial and final nodal positions; The AEDD solution is obtained from using autoencoders trained with a material dataset of size $M = 40^3$.

The trained autoencoders corresponding to different material datasets and architetures are then applied to data-driven simulations, where the number of nearest neighbors used in locally convex reconstruction of the data-driven solver is set as 6. *NRMSD* of data-driven solutions with respect to the model-based reference solution is given in the bottom row of Fig. 6.6. For all architectures, it can be observed that the AEDD solutions (both mean values and variation) improve as the number of training data increases, which suggests a good convergence property. Although using an embedding dimension of 5 (encoder: $6-10-8-5$) yields the highest accuracy, the AEDD solutions obtained from using an embedding dimension of 3 and 4 are satisfactory. It also shows that the overall patterns of error convergence in *NRMSDs* are similar across different encoder architectures using the same size of training dataset, indicating that the AEDD solutions are not sensitive to the width and depth of the encoder architecture as long as autoencoders of a sufficiently large size are used. Considering that using a more complex encoder architecture with a larger embedding dimension would increase computational cost in



(a)



(b)

Figure 6.8. Comparison of LCDD and AEDD: (a) Number of iterations against number of training data; (b) normalized computational time against number of training data. The noisy datasets and the encoder architecture of 6-4-3 are employed in this test.

data-driven computing, an encoder architecture $6-4-3$ is used in the numerical examples.

Fig. 6.7 shows that the normalized tip deflection-loading curve predicted by the proposed AEDD method agrees well with the model-based reference. The noisy data set of size $M = 40^3$ is used in this case. The results obtained by LCDD are also provided for comparison in Fig. 6.7(a), where a few loading steps yield divergent data-driven solutions when the noisy material data is employed. On the other hand, the AEDD method stays robust even with noisy data employed. It is also worth noting that when using Solver I (Eq. (6.5)) in AEDD, we also observe unconverged solutions (which are not reported in the Figures). We attribute this to the information loss caused by the decoder functions. On the other hand, Solver II with the convex interpolation functions defined in the embedding space and the material data points in data space yields stable solutions.

The comparison of AEDD and LCDD with respect to the iteration number and the computational cost are given in Fig. 6.8. In this case, the architecture of $6-4-3$ is used. While the number of iterations for convergence varies in AEDD due to the non-uniqueness of autoencoder training, it generally requires less data-driven iterations than LCDD to achieve converged solutions, as shown in Fig. 6.8(a). This is because a more generalized embedding space is used in AEDD for computing the local material solution. We also observe that with less noisy material data, the required iteration number decreases regardless of the increase in data size, an attractive property for data-driven computing. Moreover, because the data search and the convexity-preserving interpolation in AEDD local solver are performed in the low-dimensional embedding space instead of the high-dimensional data space, the computational cost of AEDD is substantially reduced compared to LCDD, as shown in Fig. 6.8(b).

Data-driven modeling with sparse noisy datasets

To evaluate the performance of the proposed AEDD approach when datasets are sparse, three noisy material datasets (Table 6.1) are generated in a similar manner as described in Section 6.4.1 but with fewer data points compared to Fig. 6.5. First, several loading paths are selected with uniformly distributed Green-Lagrangian strains for each of the loading paths. The corresponding 2nd PK stresses are generated using the elastic tensor given in Eq. (6.7). Consequently, the sparse noisy material datasets are given in Fig. 6.9, where Gaussian perturbations scaled by $0.4\bar{\mathbf{z}}_{max}/\sqrt[3]{M}$ are added independently pointwise to both the strain and the stress data.

Table 6.1. Sparse material datasets

Sparse Dataset	Number of Loading Paths	Number of Data Points per Loading Path	Total Number of Data Points (M)
1	56	10	560
2	98	10	980
3	98	8	784

An autoencoder (6-4-3) is trained using the sparse noisy datasets and used in AEDD modeling of the cantilever beam problem. The normalized tip deflection-loading responses predicted by the proposed AEDD method are compared with the constitutive model-based solutions, as shown in Fig. 6.10. The results demonstrate that the proposed AEDD method remains robust and accurate when dealing with noisy material datasets at different levels of sparsity and that the data-driven prediction accuracy improves as the data density increases.

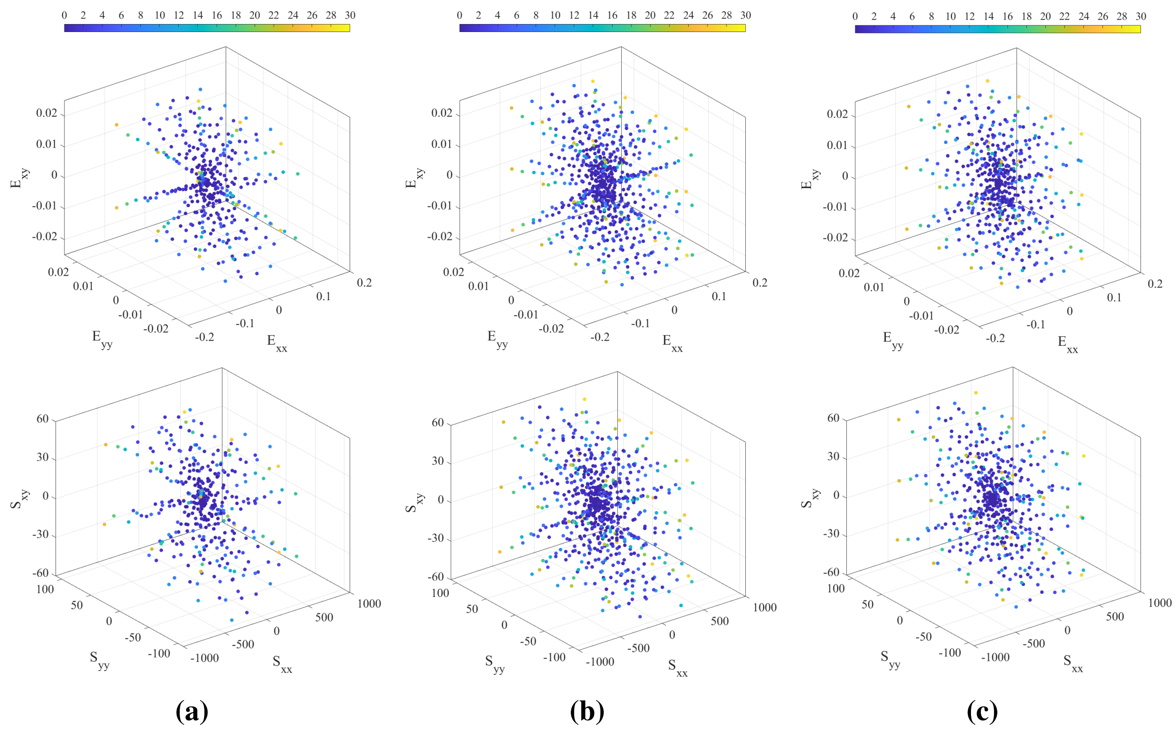


Figure 6.9. Sparse noisy material datasets: (a) sparse dataset 1; (b) sparse dataset 2; (c) sparse dataset 3; Top: strain components; Bottom: stress components

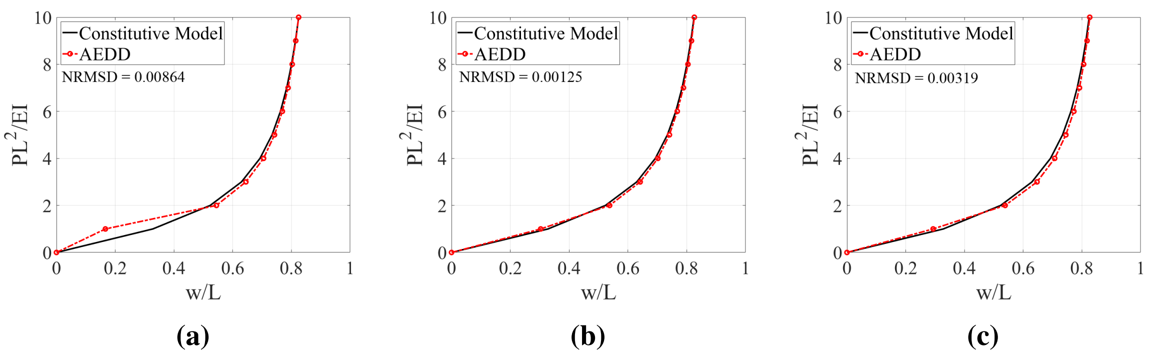


Figure 6.10. Comparison of constitutive model-based and AEDD solutions: (a) sparse dataset 1; (b) sparse dataset 2; (c) sparse dataset 3

6.4.2 Biological tissue data-driven modeling

The effectiveness of the proposed AEDD computational framework is examined by using the biological data from biaxial mechanical experiments of a porcine mitral valve posterior leaflet (MVPL) [210]. Fig. 6.11(a) shows the schematic of a MVPL specimen with a dimension $7.5mm \times 7.5mm$ subjected to prescribed displacements, where the tissue's circumferential and radial directions are denoted as x and y axes, respectively, and the stretch ratios along these two directions are defined as λ_{Circ} and λ_{Rad}.

A total of eleven protocols (Table 6.2) includes *nine biaxial tension protocols* with various tension ratios and *two pure shear protocols*, as illustrated in Fig. 6.11(b). The normal components of the Green strain and the associated 2nd-PK stress tenors generated from the 11 biaxial mechanical testing are plotted in Fig. 6.11(d) and Fig. 6.11(e), respectively. It shows that the measured data points are sparse in the stress-strain phase space. It is noted that in the mechanical testing the direct measurements are the applied membrane tensions, T_{Rad} and T_{Circ}, and the displacements are estimated by digital image correlation techniques. Thus, the measured Green strain and 2nd-PK stress data are based on homogeneous deformation assumption in the test specimen. More details about the tissue strain and stress calculations as well as the experimental setting can be found in [47, 210].

Five study cases are considered to evaluate the performance of the proposed AEDD framework, which is compared with that of the LCDD method [3, 47]. In these tests (Case 1–5), the experimental data (see Fig. 6.11(d-e)) associated with the selected biaxial testing protocols, called *training protocols*, are used for constructing material dataset $\mathbb{E}$, and different data-driven modeling approaches with the constructed material dataset are tested on other protocols (called *testing protocols*) to assess their performance against the experimental results. The training and testing protocols of the Five study cases are described as below:

- **Case 1**: *Training Protocols*: 1, 3, 4, 7, and 8; *Testing Protocols*: 2 and 5, used to investigate AEDD's performance in interpolative and extrapolative predictions.

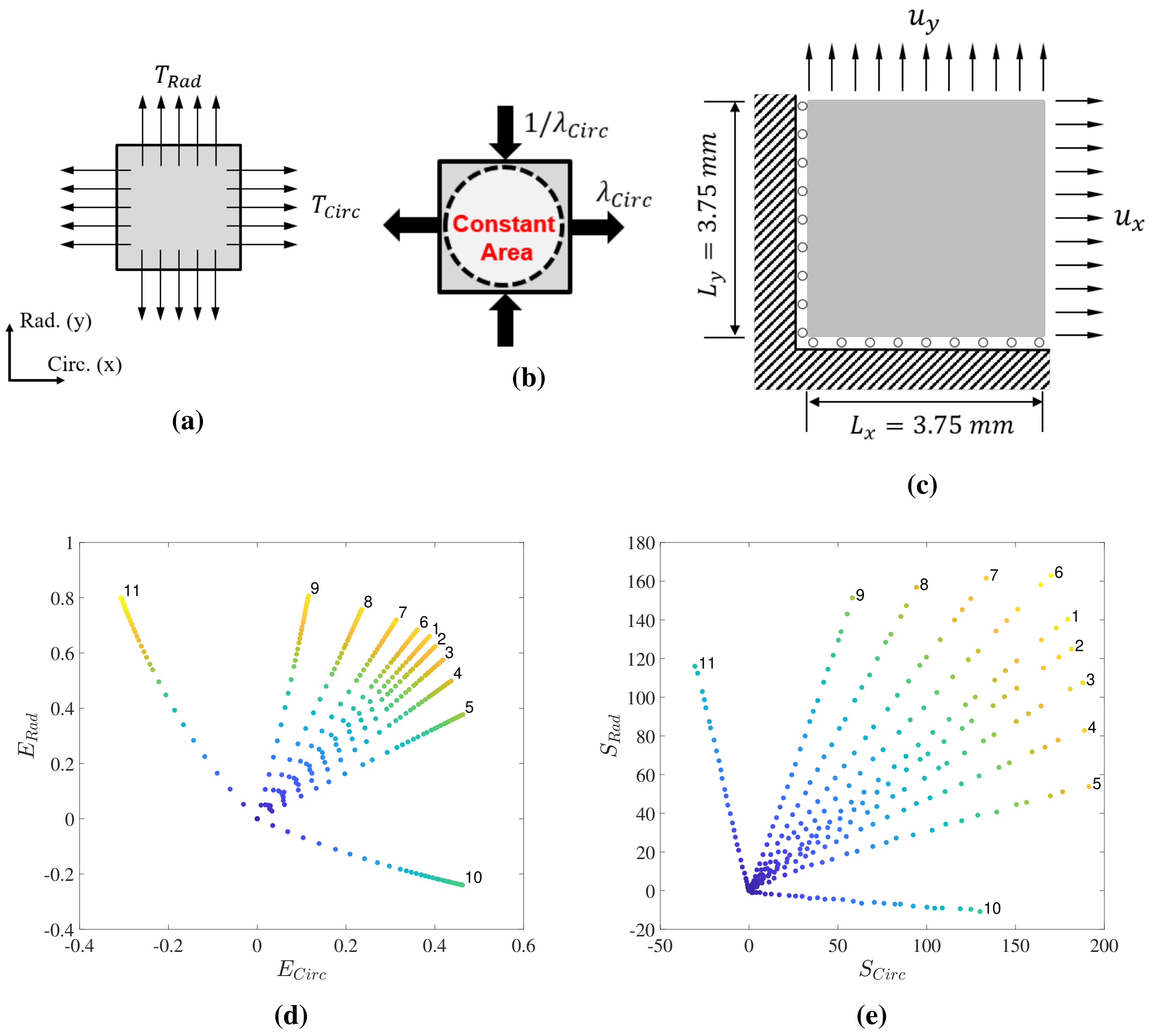


Figure 6.11. (a) Schematic of a mitral valve posterior leaflet (MVPL) specimen mounted on a biaxial testing system; (b) pure shear protocol 10 (x: tension, y: compression); (c) schematic of the model of biaxial testing in data-driven computation; (d) Green strain of all protocols; (e) 2nd-PK stress of all protocols

Table 6.2. Eleven biaxial mechanical testing protocols of a representative MVPL specimen and the corresponding measured displacements used in data-driven computations

Protocol ID	Protocol	λ_{Circ}	λ_{Rad}	u_{Circ} (mm)	u_{Rad} (mm)
1	Biaxial Tension T_{Circ} : T_{Rad}=1:1	1.333	1.525	2.498	3.938
2	Biaxial Tension T_{Circ} : T_{Rad}=1:0.8	1.342	1.499	2.564	3.744
3	Biaxial Tension T_{Circ} : T_{Rad}=1:0.6	1.355	1.466	2.662	3.498
4	Biaxial Tension T_{Circ} : T_{Rad}=1:0.4	1.369	1.415	2.770	3.110
5	Biaxial Tension T_{Circ} : T_{Rad}=1:0.2	1.388	1.326	2.913	2.442
6	Biaxial Tension T_{Circ} : T_{Rad}=0.8:1	1.313	1.541	2.344	4.055
7	Biaxial Tension T_{Circ} : T_{Rad}=0.6:1	1.275	1.562	2.064	4.215
8	Biaxial Tension T_{Circ} : T_{Rad}=0.4:1	1.213	1.588	1.596	4.409
9	Biaxial Tension T_{Circ} : T_{Rad}=0.2:1	1.109	1.618	0.820	4.635
10	Pure Shear in x	1.387	0.721	2.903	-2.093
11	Pure Shear in y	0.620	1.612	-2.847	4.590

- **Case 2**: *Training Protocols*: 1, 3, 4, 7, 8, 10, and 11; *Testing Protocols*: 2 and 5, used to investigate AEDD's performance in interpolative and extrapolative predictions.
- **Case 3**: *Training Protocols*: 1, 2, 6, 10, and 11; *Testing Protocols*: 3 and 4, used to investigate AEDD's performance in extrapolative prediction.
- **Case 4**: *Training Protocols*: 2, 5, 7, and 8, which are asymmetrically distributed; *Testing Protocols*: 1, 3 and 4, used to investigate AEDD's performance in intrapolative prediction.
- **Case 5**: *Training Protocols*: 1 – 9; *Testing Protocols*: 10 and 11.

For the first three cases, the protocols used for training are symmetrically distributed, while the training protocols are asymmetrically distributed for the last case.

For AEDD, autoencoders are first trained offline using the training protocols and then employed in the local step of the data-driven solvers (Section 6.3.2) of AEDD during the online computation. In the following study, Solver II (Section 6.3.2) is employed and the number of nearest neighbors in locally convex reconstruction of the data-driven solver is set as 6. A diagonal matrix is used as the weight matrix $\hat{\mathbb{C}}$ with each diagonal component being the ratio of the standard deviation of the associated component of the stress data to that of the strain data.

This is similar to the normalization technique used in deep learning that applies the standard deviation of each input unit to inversely scale the input data [27].

The prediction of data-driven methods on testing protocols that are not included in the training dataset are compared with the corresponding experimental data. The *NRMSD* (Eq. (6.8)) normalized with respect to the maximum stress of the experimental data is employed to assess the prediction performance of the methods. In the data-driven modeling, considering the symmetric geometry of the tissue specimen and the symmetric loading conditions, the upper right quarter of the sample is modelled with symmetric boundary conditions, as shown in Fig. 6.11(c), and the prescribed displacements are applied to the top and the right boundaries.

Case 1

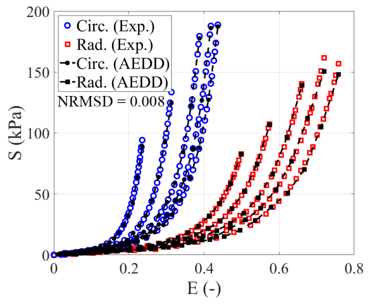
We first examine the data fitting capability whereby the data-driven methods are tested on the training protocols 1, 3, 4, 7 and 8, as shown in Fig. 6.12(a) and Fig. 6.12(d). It shows that both AEDD ($NRMSD_{AEDD}$=0.008) and LCDD ($NRMSD_{LCDD}$=0.022) provide satisfactory fitting results, but AEDD yields a slightly higher accuracy. Since the strains and stresses of testing protocols 2 and 5 lie inside and outside the domain covered by the data of the training protocols, respectively, as shown in Fig. 6.11(d-e), the AEDD predictions on the testing protocols 2 and 5 are interpolative and extrapolative predictions, respectively. For the interpolative prediction test on Protocol 2, the results of these two approaches also agree well with the experimental data, as shown in Fig. 6.12(b) and (e). The *NRMSD* errors indicate that LCDD achieves a higher accuracy, i.e. $NRMSD_{LCDD} = 0.009 < NRMSD_{AEDD} = 0.021$. However, its extrapolative prediction on Protocol 5 is worse than that from AEDD ($NRMSD_{LCDD} = 0.158 > NRMSD_{AEDD} = 0.059$), see Fig. 6.12(c) and (f). The results demonstrate better extrapolative generalization ability of AEDD. It could be attributed to the underlying low-dimensional global material manifold learned by the autoencoders. Specifically, AEDD performs local neighbor searching and locally convex reconstruction of optimal material state based on geometric distance information in the low-dimensional global embedding space, which contains the underlying manifold structure of the material data and contributes to a higher solution accuracy and better generalization performance. In contrast, LCDD performs local neighbor searching and locally convex reconstruction purely from the existing material data points without any generalization, leading to lower extrapolative generalization ability.

Another proposed AEDD method with Solver I (Section 6.3.2) using the same training protocols as material dataset are also investigated, as shown in Fig. 6.13. As expected, compared to the results obtained by using Solver II, see Fig. 6.12(b) and (c), the prediction capability by Solver I decreases on both testing protocols. Especially in the interpolative prediction test Protocol 2, the *NRMSD* error increases to 0.04 from 0.021. We attribute the larger errors with Solver I to the employment of decoders in constructing the optimal material state. Since we have demonstrated that the AEDD approach with Solver II provides better data-driven prediction results, we only consider this approach in the following study.

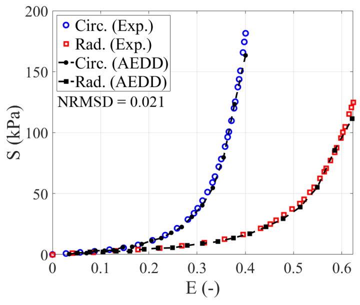
Case 2

In this case study, the objective is to verify how the incorporation of material data of different deformation modes affects the interpolative and extrapolative predictability in the proposed data-driven modeling. Two pure shear protocols are introduced in the training material dataset in addition to the biaxial tension protocols used in Case 1. The two pure shear protocols (10 and 11) in the training dataset exhibit different material behaviors from the remaining biaxial tension protocols (1, 3, 4, 7, and 8). The AEDD predictions on the testing protocols 2 and 5 are interpolative and extrapolative predictions, respectively.

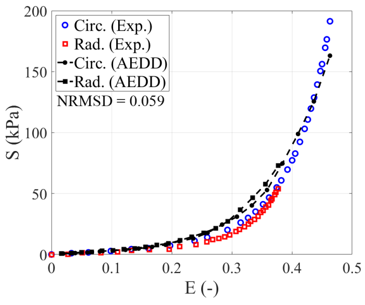
As can be seen from Fig 6.14(a) and (d), both LCDD and AEDD maintain good fitting performance for all the biaxial tension and pure shear training protocols. They also perform well for the testing Protocol 2 (Fig. 6.14(b) and (e)) with almost the same accuracy in Case 1. This is a desirable property in data-driven methods. AEDD again yields higher accuracy than LCDD in the extrapolative test (Protocol 5), as evidenced by the smaller NRMSD value 0.071 in the AEDD prediction over 0.159 in the LCDD prediction. This further demonstrates the enhanced



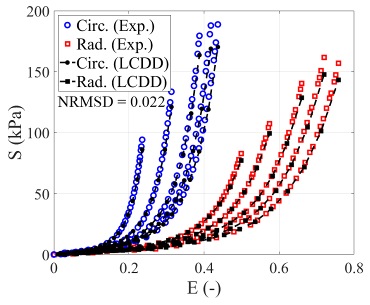
(a) Protocols 1, 3, 4, 7, 8



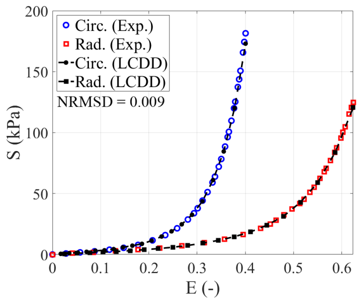
(b) Protocol 2



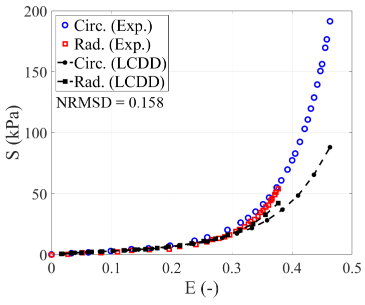
(c) Protocol 5



(d) Protocols 1, 3, 4, 7, 8

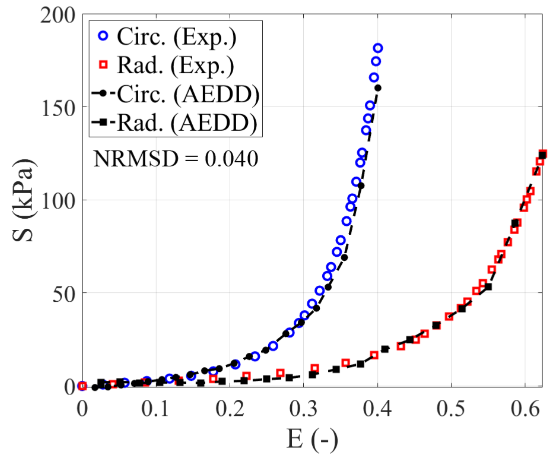


(e) Protocol 2

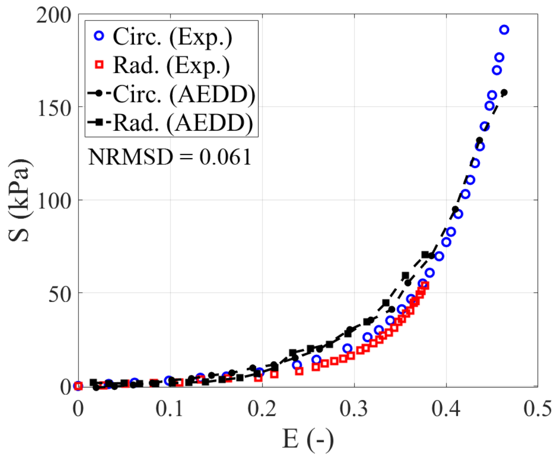


(f) Protocol 5

Figure 6.12. Comparison of interpolative (Protocol 2) and extrapolative (Protocol 5) predictability: (a) AEDD prediction on training Protocols 1, 3, 4, 7, 8; (b) AEDD prediction on Protocol 2; (c) AEDD prediction on Protocol 5; (d) LCDD prediction on training Protocols 1, 3, 4, 7, 8; (e) LCDD prediction on Protocol 2; (f) LCDD prediction on Protocol 5. Protocols 1, 3, 4, 7, and 8 are used to train the autoencoder applied in AEDD



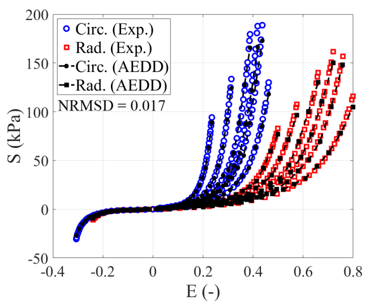
(a) Protocol 2



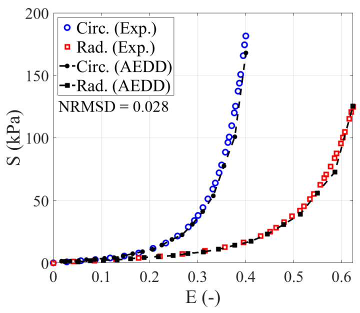
(b) Protocol 5

Figure 6.13. Data-driven prediction by AEDD with Solver I on (a) Protocol 2 and (b) Protocol 5. Protocols 1, 3, 4, 7, and 8 are used to train the autoencoder

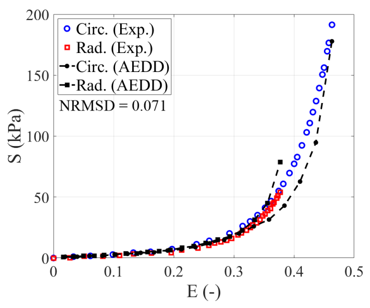
extrapolative generalization in the proposed autoencoder-based approach. Moreover, compared with Case 1, Fig. 6.14(c) shows that AEDD with the material data from the pure shear protocols improves the prediction for strain $E < 0.35$ but results in slightly more discrepancies in the high strain range.



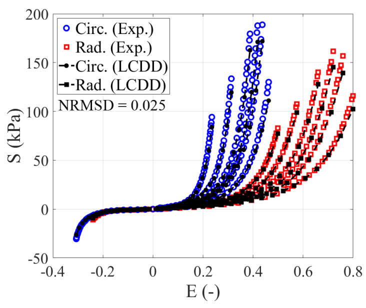
(a) Protocols 1, 3, 4, 7, 8, 10, 11



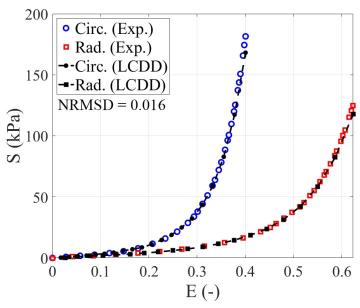
(b) Protocol 2



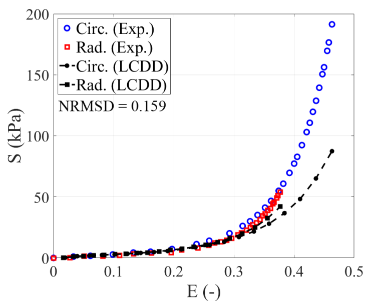
(c) Protocol 5



(d) Protocols 1, 3, 4, 7, 8, 10, 11



(e) Protocol 2

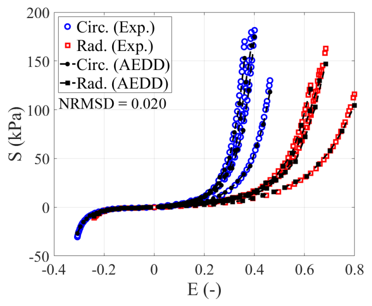


(f) Protocol 5

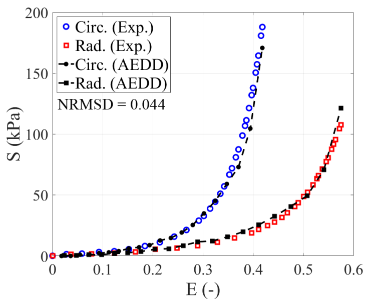
Figure 6.14. Comparison of interpolative (Protocol 2) and extrapolative (Protocol 5) predictability: (a) AEDD prediction on training Protocols 1, 3, 4, 7, 8, 10, 11; (b) AEDD prediction on Protocol 2; (c) AEDD prediction on Protocol 5; (d) LCDD prediction on training Protocols 1, 3, 4, 7, 8, 10, 11; (e) LCDD prediction on Protocol 2; (f) LCDD prediction on Protocol 5. Protocols 1, 3, 4, 7, 8, 10, and 11 are used to train the autoencoder applied in AEDD

Case 3

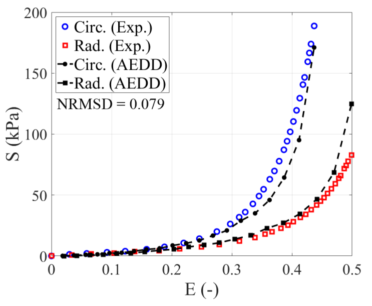
The extrapolative prediction performance of AEDD is further explored in this case study. Here, three biaxial tension protocols (Protocols 1, 2, and 6) with similar loading patterns, as illustrated by the experimental data in Fig. 6.11(d) and (e), and two pure shear protocols (Protocols 10 and 11) are used for the material training dataset. The AEDD and LCDD approaches are tested on two testing protocols (Protocols 3 and 4) subjected to larger loading ratio differences between tissue's circumferential and radial directions. Again, Fig. 6.15 shows that AEDD outperforms LCDD in both training and testing protocols. In the testing cases (Protocols 3 and 4), while the LCDD results show clear discrepancies from the experimental data, AEDD provides a better accuracy, as evidenced by reducing the NRMSD with more than 50% from the LCDD prediction. This example further verifies better extrapolative generalization capability of AEDD.



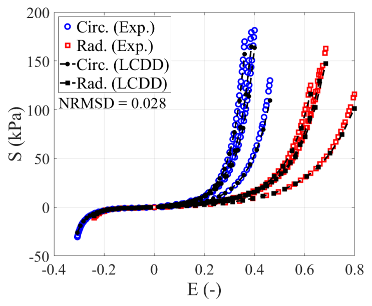
(a) Protocols 1, 2, 6, 10, 11



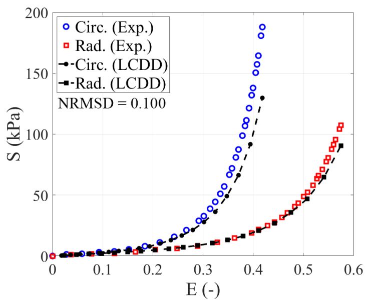
(b) Protocol 3



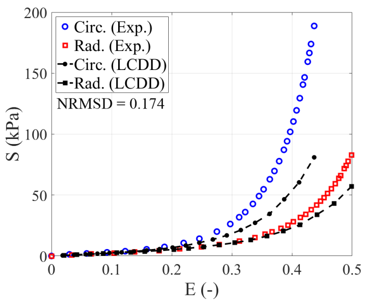
(c) Protocol 4



(d) Protocols 1, 2, 6, 10, 11



(e) Protocol 3



(f) Protocol 4

Figure 6.15. Comparison of extrapolative predictability: (a) AEDD prediction on training Protocols 1, 2, 6, 10, 11; (b) AEDD prediction on Protocol 3; (c) AEDD prediction on Protocol 4; (d) LCDD prediction on training Protocols 1, 2, 6, 10, 11; (e) LCDD prediction on Protocol 3; (f) LCDD prediction on Protocol 4. Protocols 1, 2, 6, 10, and 11 are used to train the autoencoder applied in AEDD

Case 4

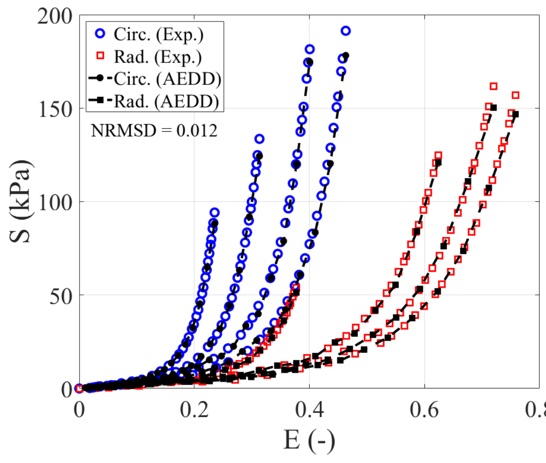
As can be seen from Fig. 6.12 and 6.14, both LCDD and AEDD work well for interpolative testing cases when using training protocols with symmetrical loading conditions. In Case 4, we investigate how a material training dataset from asymmetrically distributed protocols (biaxial tension Protocols 2, 5, 7, and 8) affects the interpolative prediction performance. Although the simulation results on the training protocols from both AEDD and LCDD agree well with experimental data, as shown in Fig. 6.16(a) and (b), the accuracy of LCDD deteriorates substantially on the testing protocols compared to AEDD, as shown in Fig. 6.16(g). The results demonstrate that AEDD's performance is more robust when dealing with irregular training datasets, which could be attributed to the underlying material manifold learned by the autoencoders.

Case 5

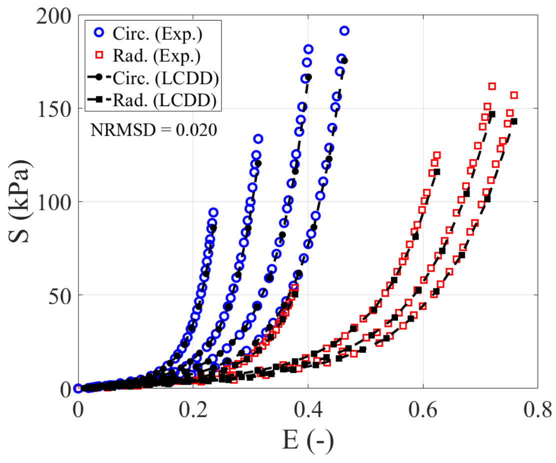
The results in Cases 1–4 have demonstrated that AEDD yields improved interpolative and extrapolative prediction compared to the LCDD approach by introducing autoencoders in the material data-driven local solver. In this last case, we investigate the performance of the AEDD method on the testing dataset that are fully unrelated to the training dataset. Specifically, autoencoders were trained using the biaxial tension protocols 1–9 for AEDD predictions on the pure shear protocols 10 and 11. As displayed in Fig. 6.17, AEDD predictions on the pure shear protocols (10 and 11) show some deviations from the experimental data. It is because the training protocols are all biaxial tension protocols that do not contain any information about the material behaviors in the pure shear protocols. These results demonstrate that the predictive capability of the machine learning techniques such as AEDD depends on the richness and quality of the given training data.

6.5 Summary

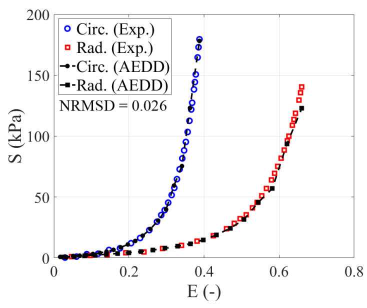
In this study, we introduced the deep manifold learning approach via autoencoders to learn the underlying material data structure and incorporated it into the data-driven solver to enhance solution accuracy, generalization ability, efficiency, and robustness in data-driven computing. The proposed approach is thus named auto-embedding data-driven (AEDD) computing. In this approach, autoencoders are trained in an offline stage and thus consume little computational overhead in solution procedures. The trained autoencoders are then applied in the proposed data-driven solver during online computation. The trained encoders and decoders define the



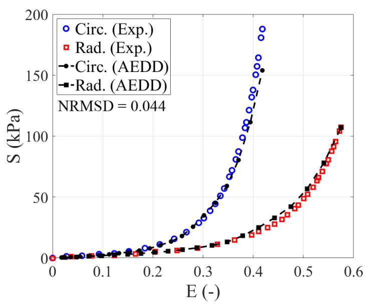
(a) Protocols 2, 5, 7, 8



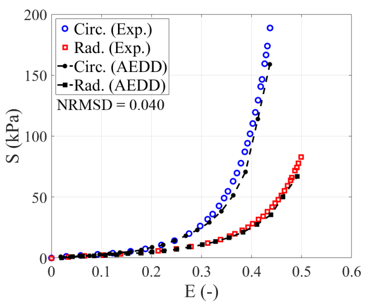
(b) Protocols 2, 5, 7, 8



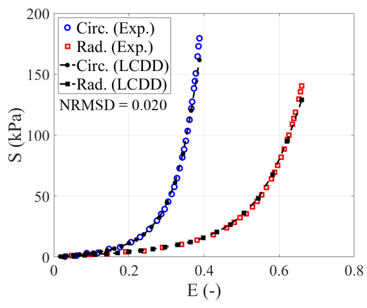
(c) Protocol 1



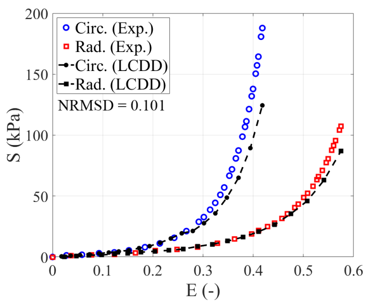
(d) Protocol 3



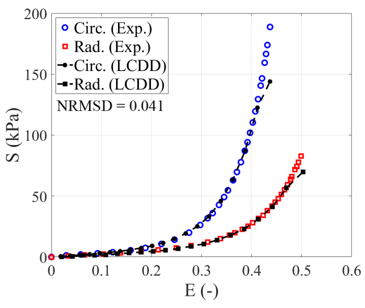
(e) Protocol 4



(f) Protocol 1

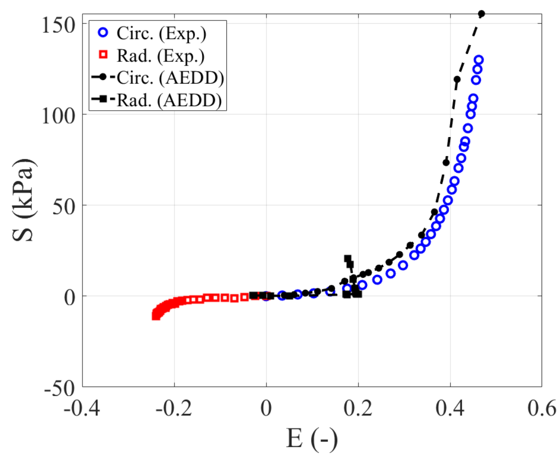


(g) Protocol 3

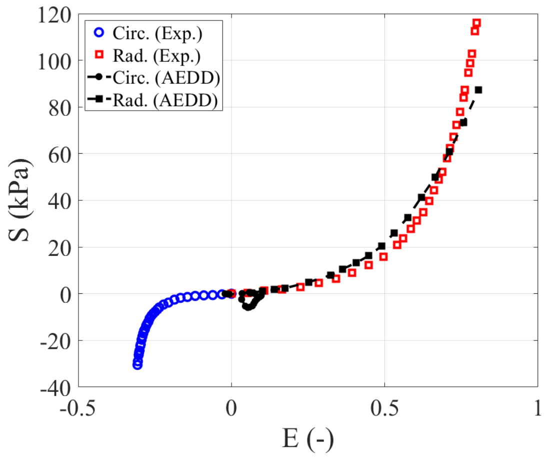


(h) Protocol 4

Figure 6.16. Comparison of interpolative predictability: (a) AEDD prediction on training Protocols 2, 5, 7, 8; (b) LCDD prediction on training Protocols 2, 5, 7, 8; (c) AEDD prediction on Protocol 1; (d) AEDD prediction on Protocol 3; (e) AEDD prediction on Protocol 4; (f) LCDD prediction on Protocol 1; (g) LCDD prediction on Protocol 3; (h) LCDD prediction on Protocol 4. Protocols 2, 5, 7, and 8 are used to train the autoencoder applied in AEDD



(a) Protocol 10



(b) Protocol 11

Figure 6.17. Data-driven prediction by AEDD on (a) Protocol 10 and (b) Protocol 11. Protocols 1–9 are used to train the autoencoder

explicit transformation between low- and high-dimensional spaces of material data, enabling efficient embedding extension to new data points. A simple Shepard convex interpolation scheme is employed in the proposed data-driven solver to preserve convexity in the local data reconstruction, enhancing the robustness of the data-driven solver.

A parametric study is conducted in the beam problem to investigate the effects of noise in material datasets, the size and sparsity of datasets, neural networks initialization during training, and autoencoder architectures on the performance of autoencdoers and data-driven solutions. Autoencoders with four different architectures are trained with synthetic noisy material datasets generated from a phenomenological model and different random initialization. The parametric study shows the performance of the offline trained autoencoders improves as the amount of training data increases regardless of the examined autoencoder architectures and neural network initialization. AEDD predictions are accurate and robust when dealing with sparse noisy datasets with the solutions converging to the constitutive model-based reference solutions as the number of material data and data density increase. In addition, with the offline trained autoencoders and efficient Shepard convex reconstruction for online computation, AEDD shows enhanced computational efficiency compared to the LCDD approach [47]. The effectiveness of the proposed framework is further examined by modeling biological tissues using experimental data. The proposed AEDD framework shows a good performance in modeling complex materials. Through five study cases, the proposed approach show stronger generalization capability and robustness than the LCDD approach [47]. This is attributed to the fact that the local neighbor searching and locally convex reconstruction in the proposed data-driven solver is based on geometric distance information in the filtered global embedding space learned by autoencoders, which contains the underlying manifold structure of the material data. The results of the last case also demonstrates the effects of richness and quality of the training data on the predictive capability of the AEDD method.

Although using 6 nearest neighbors in the locally convex reconstruction of the data-driven solver and an empirical weight matrix $\hat{\mathbb{C}}$ based on statistical information of data is sufficient for AEDD to produce accurate and robust solutions in the problems of this study, choosing the optimal number of nearest neighbors and the optimal weight matrix requires further investigations. The results of the proposed data-driven approach demonstrate the promising performance by integrating the autoencoder enhanced deep manifold learning into data-driven computing of systems with complex material behaviors.

6.6 Acknowledgement

This chapter, in part, is a reprint of the material as it appears in: "Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021". The dissertation author is the primary investigator and author of this paper.

Chapter 7

Thermodynamically Consistent Machine-Learned Internal State Variable Approach for Data-Driven Modeling of Path Dependent Materials

7.1 Introduction

Traditional constitutive modeling is based on constitutive or material laws to describe the explicit relationship among the measurable material states, e.g., stresses and strains, and internal state variables (ISVs) based on experimental observations, mechanistic hypothesis, and mathematical simplifications. However, limited data and functional form assumptions inevitably introduce errors to the model parameter calibration and model prediction. Moreover, with the pre-defined functions, constitutive laws often lack generality to capture full aspects of material behaviors [6, 47].

Path-dependent constitutive modeling typically applies models with evolving ISVs in addition to the state space of deformation [21, 22]. The ISV constitutive modeling framework has been effectively applied to model various nonlinear solid material behaviors, e.g., elasto-plasticity [23, 24], visco-plasticity [25], and material damage [26]. However, ISVs are often non-measurable, which makes it challenging to define a complete and appropriate set of ISVs for highly nonlinear and complicated materials, e.g., geomechanical materials. Further, the traditional ISV constitutive modeling approach often results in excessive complexities with high computational cost, which is undesirable in practical applications.

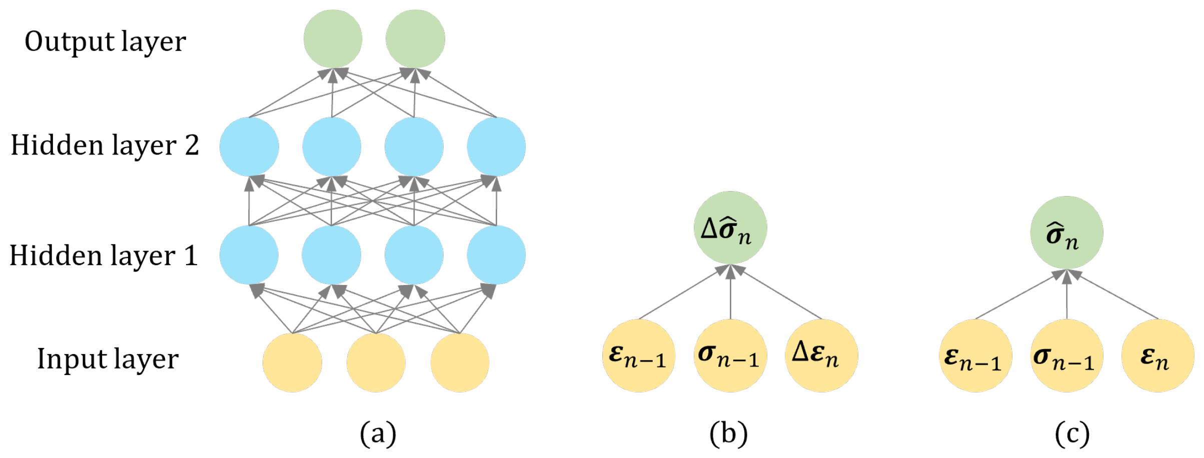
In this study, we propose a thermodynamically consistent machine-learned ISV approach for data-driven modeling of path-dependent materials, which relies purely on measurable material states. The first thermodynamics principle is integrated into the model architecture whereas the second thermodynamics principle is enforced by a constraint on the network parameters. In the proposed model, an RNN is trained to infer intrinsic ISVs from its hidden (or memory) state that captures essential history-dependent features of data through a sequential input. The RNN describing the evolution of the data-driven machine-learned ISVs follows the thermodynamics second law. In addition, a DNN is trained simultaneously to predict the material energy potential given strain, ISVs, and temperature (for non-isothermal processes). Further, model robustness and accuracy is enhanced by introducing *stochasticity* to inputs for model training to account for uncertainties of input conditions in testing.

The remainder of this chapter is organized as follows. In Section 7.2, the applications DNNs and RNNs to path-dependent materials modeling are discussed. Section 7.3 introduces the proposed thermodynamically consistent machine-learned ISV approach for data-driven modeling of path-dependent materials, where two thermodynamically consistent recurrent neural networks (TCRNNs) are discussed. Finally, in Section 7.4, the effectiveness and generalization capability of the proposed TCRNN models are examined by modeling an elasto-plastic material and undrained soil under cyclic shear loading. A parametric study is conducted to investigate the effects of the number of RNN steps, the internal state dimension, the model complexity, and the strain increment on the model performance. Concluding remarks and discussions are summarized in Section 7.5.

7.2 Black-Box Data-Driven Modeling of Path-Dependent Materials

7.2.1 Deep Neural Networks (DNNs) Constitutive Models

Deep neural networks (DNNs), as the core of the deep learning [27], represent complex models that relate data inputs, $\mathbf{x} \in \mathbb{R}^{d_{in}}$, to data outputs, $\mathbf{y} \in \mathbb{R}^{d_{out}}$. The basic concepts of deep neural networks (DNNs) have been introduced in Section 2.1.1. Fig. 7.1(a) shows the computational graph of a feed-forward DNN with three input neurons, two hidden layers, and two output neurons.



Output layer
Hidden layer 2
Hidden layer 1
Input layer
$\Delta\hat{\boldsymbol{\sigma}}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\boldsymbol{\sigma}_{n-1}$
$\Delta\boldsymbol{\varepsilon}_n$
$\hat{\boldsymbol{\sigma}}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\boldsymbol{\sigma}_{n-1}$
$\boldsymbol{\varepsilon}_n$
(a)
(b)
(c)

Figure 7.1. Computational graphs of (a) a feed-froward deep neural network (DNN) with three input neurons, two hidden layers, and two output neurons, (b) a DNN constitutive model that takes one history step of stress-strain states and the current strain increment as input and predicts the current stress increment, and (c) a DNN constitutive model that takes one history step of stress-strain states and the current total strain as input and predicts the current total stress.

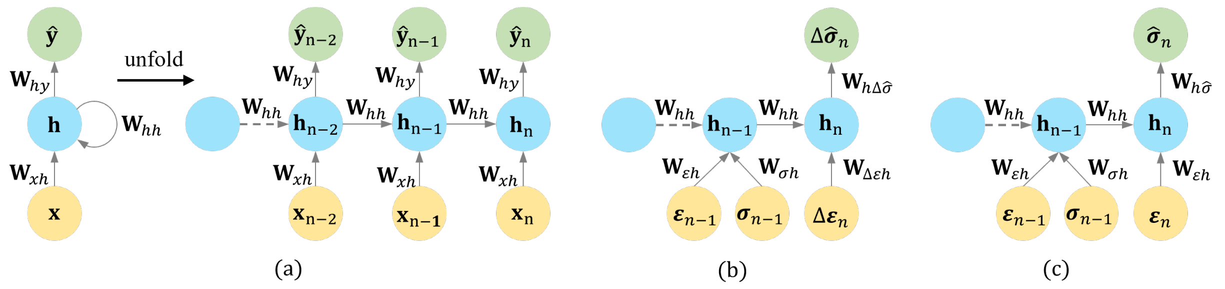
For path-dependent materials, the current stress response depends on the past stress-strain history. Fig. 7.1(b) shows the computational graph of a DNN constitutive model for path-dependent materials that takes one history step of stress-strain states and the current strain increment as input and predicts the current stress increment. Alternatively, the current strain increment can be replaced with the current total strain as input and the current total stress as the output, as shown in Fig. 7.1(c). The effects of the input/output representation will be discussed in the next subsection. These DNN constitutive models can be extended to consider pre-defined ISVs as input in addition to the measurable material states. An additional DNN can be used to model the evolution of the ISVs [80]. However, the dependency on pre-defined ISVs limits its applications, especially when only the measurable material states of path-dependent behaviors are available, e.g., the soil example to be demonstrated in Section 7.4.2.

Note that for DNN constitutive models, the number of history input steps is fixed once the model architecture is determined, which means the number of history input steps used for testing must be the same as that used in training. Furthermore, it is difficult for DNNs with recurrent connections from the output of step n to the input of step $n+1$ to capture the essential information about the past history since the outputs are explicitly trained only to match the training set targets not being informed of the past history [27]. These issues are addressed by RNNs introduced in the next subsection.

7.2.2 Recurrent Neural Networks (RNNs) Constitutive Models

Recurrent neural networks (RNNs) designed for sequence learning have demonstrated successful applications in various domains, such as machine translation and speech recognition, due to their capability of learning history-dependent features that are essential for sequential prediction [41, 42]. The basic concepts of RNNs have been introduced in Section 2.1.1. Fig. 7.2(a) illustrates the computational graphs of a folded RNN and an unfolded RNN, where $\mathbf{h}$ is a hidden state that captures essential history-dependent features from past information, which makes RNNs particularly suitable for modeling path-dependent material behaviors. Unfolding of the RNN computational graph results in *parameter sharing* across the network structure, reducing the number of trainable parameters and thus leading to more efficient training.

Depending on applications, RNNs can have flexible architectures of input and output, such as one-to-one, one-to-many, many-to-one, and many-to-many [43]. For example, the unfolded RNN shown in Fig. 7.2(a) is a many-to-many type of RNN, which can be applied to, for example, name entity recognition. For path-dependent constitutive modeling, the many-



$\hat{\mathbf{y}}$
unfold
$\mathbf{W}_{hy}$
$\mathbf{h}$
$\mathbf{W}_{hh}$
$\mathbf{W}_{xh}$
$\mathbf{x}$
$\hat{\mathbf{y}}_{n-2}$
$\hat{\mathbf{y}}_{n-1}$
$\hat{\mathbf{y}}_{n}$
$\mathbf{h}_{n-2}$
$\mathbf{h}_{n-1}$
$\mathbf{h}_{n}$
$\mathbf{x}_{n-2}$
$\mathbf{x}_{n-1}$
$\mathbf{x}_{n}$
(a)
$\Delta\hat{\boldsymbol{\sigma}}_n$
$\mathbf{W}_{h\Delta\hat{\sigma}}$
$\mathbf{W}_{\varepsilon h}$
$\mathbf{W}_{\sigma h}$
$\mathbf{W}_{\Delta\varepsilon h}$
$\boldsymbol{\varepsilon}_{n-1}$
$\boldsymbol{\sigma}_{n-1}$
$\Delta\boldsymbol{\varepsilon}_n$
(b)
$\hat{\boldsymbol{\sigma}}_n$
$\mathbf{W}_{h\hat{\sigma}}$
$\boldsymbol{\varepsilon}_n$
(c)

Figure 7.2. Computational graphs of (a) a recurrent neural network (RNN), (b) a RNN constitutive model that takes one history step of stress-strain states and the current strain increment as input and predicts the current stress increment, and (c) an RNN constitutive model that takes one history step of stress-strain states and the current total strain as input and predicts the current total stress.

to-one type of RNN is more suitable. Fig. 7.2(b) illustrates the computational graph of an RNN constitutive model that takes one history step of stress-strain states and the current strain increment as input and predicts the current stress increment, defined as an *incremental* RNN, whereas Fig. 7.2(c) show the computational graph of an RNN constitutive model that takes one history step of stress-strain states and the current total strain as input and predicts the current total stress, defined as a *total-form* RNN. Unlike the standard RNNs that has the same input size for all time steps (states), the history-step input of the RNN constitutive models shown in Fig. 7.2(b)-(c) contains both strain and stress components whereas the current-step input contains only strain components. Considering one history step, the forward propagation of a typical *total-form* RNN is expressed as follows

$$\mathbf{h}_{n-1} = a_{tanh}\left(\mathbf{W}_{hh}\mathbf{h}_{n-2} + \mathbf{W}_{\varepsilon h}\boldsymbol{\varepsilon}_{n-1} + \mathbf{W}_{\sigma h}\boldsymbol{\sigma}_{n-1} + \mathbf{b}_h\right), \tag{7.1a}$$

$$\mathbf{h}_n = a_{tanh}\left(\mathbf{W}_{hh}\mathbf{h}_{n-1} + \mathbf{W}_{\varepsilon h}\boldsymbol{\varepsilon}_n + \mathbf{b}_h\right), \tag{7.1b}$$

$$\hat{\boldsymbol{\sigma}}_n = \mathbf{W}_{h\hat{\sigma}}\mathbf{h}_n + \mathbf{b}_y, \tag{7.1c}$$

where $\mathbf{W}_{hh}$, $\mathbf{W}_{\varepsilon h}$, $\mathbf{W}_{\sigma h}$, and $\mathbf{W}_{h\hat{\sigma}}$ are trainable weight coefficients for hidden-to-hidden, strain-to-hidden, stress-to-hidden, and hidden-to-output transformations, respectively, and $\mathbf{b}_h$ and $\mathbf{b}_y$ are trainable bias coefficients.

To capture complex history-dependent patterns, deep RNNs are more advantageous [27]. Similar to DNNs, stacking of fully-connected hidden layers can be added to affine input-to-hidden, hidden-to-hidden, and hidden-to-output transformations.

Remark. *Our studies show that the total-form RNN is less sensitive to the strain increment than the incremental RNN, as a consequence of* ***interpolation*** *outperforming* ***extrapolation****. For instance, considering a training dataset with one stress-strain path and a constant loading (strain) increment, the final-step input (the strain increment) of the incremental RNN is constant, whereas the final-step input (the total strain) of the total-form RNN is not constant and covers the whole range of strain in the stress-strain path. During testing on the same stress-strain path but with a different constant loading increment, larger or smaller than that of the training data, the incremental RNN becomes inaccurate since the final-step input (the strain increment) of the testing data is beyond the range of the training strain increment and the prediction is an* ***extrapolation****. In contrast, the total-form RNN remains accurate because the final-step input (the total strain) of the testing data is within the range of training total strain and the prediction is an* ***interpolation****. Therefore, the proposed data-driven models in this study are built upon the total form.*

Gated Recurrent Units (GRUs)

Standard RNNs suffer from short-term memory due to vanishing and exploding gradient issues that arise from recurrent connections [27, 211, 212]. More effective RNNs for learning long-term dependencies have been developed, including the long short-term memory (LSTM) [81] cells and gated recurrent units (GRUs) [82, 83]. A typical GRU consists of a *reset* gate $\mathbf{r}_n$ that removes irrelevant past information, an *update* gate $\mathbf{u}_n$ that controls the amount of past information passing to the next step, and a *candidate* hidden state $\tilde{\mathbf{h}}_n$ [83]. Compared to LSTM, the GRU has fewer parameters as it lacks an *output* gate. Considering one history step, the forward propagation of a typical GRU is expressed as follows

$$\mathbf{r}_n = a_\sigma\big(\mathbf{W}_{hr}\mathbf{h}_{n-1} + \mathbf{W}_{xr}\mathbf{x}_n + \mathbf{b}_r\big), \tag{7.2a}$$

$$\mathbf{u}_n = a_\sigma\big(\mathbf{W}_{hu}\mathbf{h}_{n-1} + \mathbf{W}_{xu}\mathbf{x}_n + \mathbf{b}_u\big), \tag{7.2b}$$

$$\tilde{\mathbf{h}}_n = a_{tanh}\big(\mathbf{r}_n * \mathbf{W}_{h\tilde{h}}\mathbf{h}_{n-1} + \mathbf{W}_{x\tilde{h}}\mathbf{x}_n + \mathbf{b}_{\tilde{h}}\big), \tag{7.2c}$$

$$\mathbf{h}_n = \mathbf{u}_n * \mathbf{h}_{n-1} + (1 - \mathbf{u}_n) * \tilde{\mathbf{h}}_n + \mathbf{b}_h, \tag{7.2d}$$

$$\hat{\mathbf{y}}_n = \mathbf{W}_{hy}\mathbf{h}_n + \mathbf{b}_y, \tag{7.2e}$$

where $*$ denotes the Hadamard (element-wise) product; a_σ is the sigmoid function; a_{tanh} is the hyperbolic tangent function; $\mathbf{W}_{hr}$, $\mathbf{W}_{xr}$, $\mathbf{W}_{hu}$, $\mathbf{W}_{xu}$, $\mathbf{W}_{h\tilde{h}}$, $\mathbf{W}_{x\tilde{h}}$, and $\mathbf{W}_{hy}$ are trainable weight coefficients; $\mathbf{b}_r$, $\mathbf{b}_u$, $\mathbf{b}_{\tilde{h}}$, $\mathbf{b}_h$, and $\mathbf{b}_y$ are trainable bias coefficients. Eq. (7.2d) calculates the current hidden state $\mathbf{h}_n$ by a linear interpolation between the previous hidden state $\mathbf{h}_{n-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_n$, based on the update gate $\mathbf{u}_n$. The RNN-based constitutive models proposed in this study are applicable to all types of RNNs for complicated path-dependent material behaviors with long-term dependent features. In the following examples of this study, the GRUs are employed in the proposed thermodynamically consistent RNN constitutive models.

Model Training

Since the forward propagation (Eq. (2.4) and Eq. (7.2)) is inherently sequential, i.e., each time step can only be computed after the previous one, the computation of the gradient of the loss function with respect to the trainable parameters cannot be parallelized and it needs to follow the reverse unfolded computational graph. The back-propagation through time algorithm is applied to RNNs [27].

During training, the model receives the ground truth stress data of history steps, which is a *teacher forcing* procedure emerging from the maximum likelihood criterion [27]. However, the disadvantage of teacher forcing training arises when the trained model is applied in an *open-loop* test mode with the network's previous outputs fed back as input for future predictions. The computational graphs of the test mode are shown in Fig. 7.3 corresponding to the RNN models (in the train mode) shown in Fig. 7.2(b)-(c). In this case, the inputs the trained model receives could be quite different from those received during training, forcing the model to perform *extrapolative* predictions and thus lead to large errors. Furthermore, such prediction errors could occur at the very first prediction, accumulate and propagate quickly, and contaminate the subsequent predictions. To mitigate the issue of error propagation due to the teacher forcing training and enhance model accuracy and robustness, we introduce *stochasticity* to the training set by adding random perturbations to the ground truth stress data. In this way, the network can also learn the variability of the input conditions resembling those in the test mode.

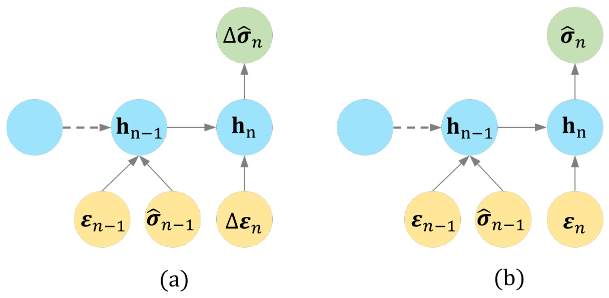

$\Delta\hat{\boldsymbol{\sigma}}_n$
$\mathbf{h}_{n-1}$
$\mathbf{h}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\hat{\boldsymbol{\sigma}}_{n-1}$
$\Delta\boldsymbol{\varepsilon}_n$
(a)
$\hat{\boldsymbol{\sigma}}_n$
$\mathbf{h}_{n-1}$
$\mathbf{h}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\hat{\boldsymbol{\sigma}}_{n-1}$
$\boldsymbol{\varepsilon}_n$
(b)

Figure 7.3. Test mode of (a) a RNN constitutive model that takes one step history of stress-strain states and the current strain increment as input and predicts the current stress increment, and (b) an RNN constitutive model that takes one history step of stress-strain states and the current total strain as input and predicts the current total stress. The stress prediction from previous step is used as a part of the history input for the current-step prediction.

7.3 Thermodynamically Consistent Machine-Learned Internal State Variable Approach for Path-Dependent Materials

7.3.1 Thermodynamically Consistent Recurrent Neural Networks (TCRNNs)

To ensure thermodynamical consistency in the data-driven path-dependent materials modeling, thermodynamics principals introduced in Section 3.3.3 are embedded into RNNs to extract the hidden ISVs. The proposed TCRNN consists of an RNN and a DNN. The computational graphs for non-isothermal or isothermal processes are illustrated in Fig. 7.4.

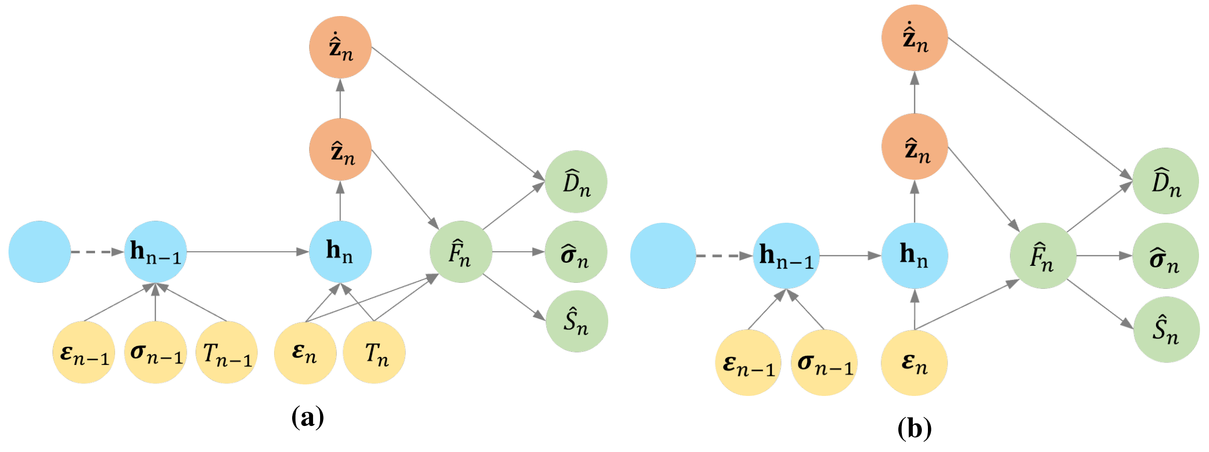

$\dot{\hat{\mathbf{z}}}_n$
$\hat{\mathbf{z}}_n$
$\hat{D}_n$
$\mathbf{h}_{n-1}$
$\mathbf{h}_n$
$\hat{F}_n$
$\hat{\boldsymbol{\sigma}}_n$
$\hat{S}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\boldsymbol{\sigma}_{n-1}$
T_{n-1}
$\boldsymbol{\varepsilon}_n$
T_n
(a)
(b)

Figure 7.4. Computational graphs of (a) a thermodynamically consistent recurrent neural network (TCRNN) for non-isothermal processes and (b) a TCRNN for isothermal processes. The rate of the machine-learned internal state variable, $\dot{\hat{\mathbf{z}}}_n$, is computed and used for calculating the dissipation rate.

Since the hidden state **h** of RNNs captures essential history-dependent features from past material information, we propose to extract ISVs of materials from the hidden state of an RNN, as shown in Fig. 7.4. Considering one history step, the RNN-inferred ISV is expressed as follows

$$\hat{\mathbf{z}}_n = f_{RNN}\left(\varepsilon_n, T_n, \varepsilon_{n-1}, \sigma_{n-1}, T_{n-1}\right). \tag{7.3}$$

Hereafter, a hat symbol is used to denote the predicted quantities. Considering one single layer for input-to-hidden, hidden-to-hidden, and hidden-to-output transformations in RNN, the RNN function f_{RNN} consists of the following

$$\mathbf{h}_{n-1} = a_h\left(\mathbf{W}_{hh}\mathbf{h}_{n-2} + \mathbf{W}_{\varepsilon h}\varepsilon_{n-1} + \mathbf{W}_{\sigma h}\sigma_{n-1} + \mathbf{W}_{Th}T_{n-1} + \mathbf{b}_h\right), \tag{7.4a}$$

$$\mathbf{h}_n = a_h\left(\mathbf{W}_{hh}\mathbf{h}_{n-1} + \mathbf{W}_{\varepsilon h}\varepsilon_n + \mathbf{W}_{Th}T_n + \mathbf{b}_h\right), \tag{7.4b}$$

$$\hat{\mathbf{z}}_n = a_z\left(\mathbf{W}_{hz}\mathbf{h}_n + \mathbf{b}_z\right), \tag{7.4c}$$

where $\mathbf{W}_{hh}$, $\mathbf{W}_{\varepsilon h}$, $\mathbf{W}_{\sigma h}$, $\mathbf{W}_{Th}$, and $\mathbf{W}_{hz}$ are trainable weight coefficients for hidden-to-hidden, strain-to-hidden, stress-to-hidden, temperature-to-hidden, and hidden-to-ISV transformations, respectively; $\mathbf{b}_h$ and $\mathbf{b}_z$ are trainable bias coefficients; $a_h(\cdot)$ and $a_z(\cdot)$ are activation functions. Note that the trainable parameters are shared across all steps of the RNN, which enhances training efficiency. $\hat{\mathbf{z}}_n$ is the machine-learned ISVs from the hidden state $\mathbf{h}_n$ of the RNN and its rate, $\dot{\hat{\mathbf{z}}}_n$, can be computed by automatic differentiation [213], which will be discussed in the next subsection. Eqs. (7.4a-b) transform the history and current measurable material states to the current hidden state $\mathbf{h}_n$ that carries the essential past information. For an RNN with more than one history step, Eq. (7.4a) is repeated for all history steps. Eq. (7.4c) infers the current ISV $\hat{\mathbf{z}}_n$ from the current hidden state $\mathbf{h}_n$.

A linear activation ($a_z(\cdot)$) is used for the transformation from the hidden state to the ISV in Eq. (7.4c). The selection of the activation $a_h(\cdot)$ requires particular attention due to the issue of second-order vanishing gradients [80]. For effective training via back-propagation, the gradient of the output derivative with respect to the trainable parameters requires non-zero second-order gradients of activation functions. As a result, the activation function $\text{SiLU}(x) = x/(1+e^{-x})$ is selected for $a_h(\cdot)$ due to its smoothness and non-zero second-order gradients, as shown in Fig. 7.5.

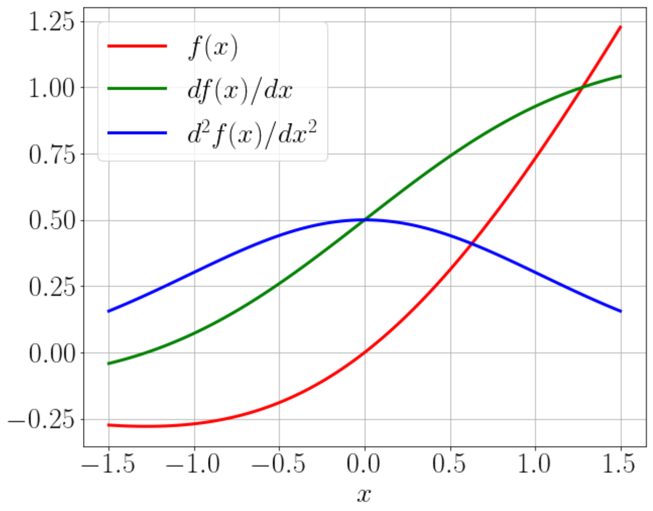


Figure 7.5. The SiLU activation function and its first-order and second-order gradients.

Following the extraction of the ISV, a DNN is then used to predict the Helmholtz free energy given the strain, the temperature and the machine-learned ISVs,

$$\hat{F}_n = f_{DNN}\big(\varepsilon_n, T_n, \hat{\mathbf{z}}_n\big), \tag{7.5}$$

where f_{DNN} represents a DNN, as discussed in Section 7.2.1. The activation in the output layer is linear. The output Helmholtz free energy is then used to compute the stress $\hat{\sigma}$, the dissipation rate $\hat{D}$, and the entropy $\hat{S}$ according to Eq. (3.31), which implicitly enforces the *first thermodynamics principle*. The *second thermodynamics principle*, i.e., $\hat{D} \geq 0$, is enforced in the loss function by constraining the network parameters, which will be discussed in the next subsection. The gradients of the output with respect to the inputs are obtained by automatic differentiation [213]. Since the output derivatives are involved in the loss function, SiLU is used for the activation of the hidden layers to avoid the issue of second-order vanishing gradients as discussed above.

7.3.2 Secondary Outputs

To balance feature contributions to the loss function and accelerate the training process, the training dataset is standardized to have zero mean and unit variance. For instance, a feature $\mathbf{x}$ of the dataset is standardized by its mean $\mu_{\mathbf{x}}$ and standard deviation $std_{\mathbf{x}}$,

$$\bar{\mathbf{x}} = \frac{\mathbf{x} - \mu_{\mathbf{x}}}{std_{\mathbf{x}}}. \tag{7.6}$$

Hereafter, a bar symbol is used to denote standardized quantities. From Eq. (7.6), we have

$$d\bar{\mathbf{x}} = \frac{1}{std_{\mathbf{x}}} d\mathbf{x}. \tag{7.7}$$

Considering standardized variables, Eqs. (7.3) and (7.5) become

$$\hat{\mathbf{z}}_n = f_{RNN}\big(\bar{\varepsilon}_n, \bar{T}_n, \bar{\varepsilon}_{n-1}, \bar{\sigma}_{n-1}, \bar{T}_{n-1}\big), \tag{7.8}$$

$$\hat{\bar{F}}_n = f_{DNN}\big(\bar{\varepsilon}_n, \bar{T}_n, \hat{\mathbf{z}}_n\big). \tag{7.9}$$

Therefore, the predicted stress is calculated by

$$\hat{\sigma}_n = \frac{\partial \hat{F}_n}{\partial \varepsilon_n} = \frac{\partial \hat{F}_n}{\partial \hat{\bar{F}}_n}\frac{\partial \hat{\bar{F}}_n}{\partial \bar{\varepsilon}_n} : \frac{\partial \bar{\varepsilon}_n}{\partial \varepsilon_n} = \frac{std_F}{std_\varepsilon}\frac{\partial \hat{\bar{F}}_n}{\partial \bar{\varepsilon}_n}. \tag{7.10}$$

with

$$\frac{\partial \hat{F}_n}{\partial \hat{\bar{F}}_n} = std_F \quad \text{and} \quad \frac{\partial \bar{\varepsilon}_n}{\partial \varepsilon_n} = \frac{1}{std_\varepsilon}\mathbf{I}, \tag{7.11}$$

according to Eq. (7.7). $\mathbf{I}$ is the second-order identity tensor. $\frac{\partial \hat{\bar{F}}_n}{\partial \bar{\varepsilon}_n}$ is the gradient of the output with respect to the input in Eq. (7.9) and can be obtained by automatic differentiation [213]. Similarly, the predicted entropy is computed by

$$\hat{S}_n = -\frac{\partial \hat{F}_n}{\partial T_n} = -\frac{\partial \hat{F}_n}{\partial \hat{\bar{F}}_n}\frac{\partial \hat{\bar{F}}_n}{\partial \bar{T}_n}\frac{\partial \bar{T}}{\partial T_n} = -\frac{std_F}{std_T}\frac{\partial \hat{\bar{F}}_n}{\partial \bar{T}_n}. \tag{7.12}$$

The predicted dissipation rate is computed by

$$\hat{D}_n = -\frac{\partial \hat{F}_n}{\partial \hat{\mathbf{z}}_n} \cdot \dot{\hat{\mathbf{z}}}_n = -\frac{\partial \hat{F}_n}{\partial \hat{\bar{F}}_n}\frac{\partial \hat{\bar{F}}_n}{\partial \hat{\mathbf{z}}_n} \cdot \dot{\hat{\mathbf{z}}}_n = -std_F \frac{\partial \hat{\bar{F}}_n}{\partial \hat{\mathbf{z}}_n} \cdot \dot{\hat{\mathbf{z}}}_n, \tag{7.13}$$

where $\dot{\hat{\mathbf{z}}}_n$ can be obtained by applying the chain rule to Eq. (7.8)

$$\dot{\hat{\mathbf{z}}}_n = \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{\varepsilon}_n} : \dot{\bar{\varepsilon}}_n + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{T}_n}\dot{\bar{T}}_n + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{\varepsilon}_{n-1}} : \dot{\bar{\varepsilon}}_{n-1} + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{\sigma}_{n-1}} : \dot{\bar{\sigma}}_{n-1} + \frac{\partial \hat{\mathbf{z}}_{n-1}}{\partial \bar{T}_n}\dot{\bar{T}}_{n-1}, \tag{7.14a}$$

$$= \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{\varepsilon}_n} : \frac{\dot{\varepsilon}_n}{std_\varepsilon} + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{T}_n}\frac{\dot{T}_n}{std_T} + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{\varepsilon}_{n-1}} : \frac{\dot{\varepsilon}_{n-1}}{std_\varepsilon} + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{\sigma}_{n-1}} : \frac{\dot{\sigma}_{n-1}}{std_\sigma} + \frac{\partial \hat{\mathbf{z}}_n}{\partial \bar{T}_{n-1}}\frac{\dot{T}_{n-1}}{std_T}, \tag{7.14b}$$

which requires the rate of the input variables, including $\dot{\varepsilon}_n$, $\dot{T}_n$, $\dot{\varepsilon}_{n-1}$, $\dot{\sigma}_{n-1}$, $\dot{T}_{n-1}$, etc.

The direct calculation of $\dot{\hat{\mathbf{z}}}_n$ through Eq. (7.14) can be computationally intractable, especially when the dataset size, the internal state dimension, and the number RNN input steps are large. Alternatively, the rate of the ISVs can be approximated by $\dot{\hat{\mathbf{z}}}_n \approx \Delta\hat{\mathbf{z}}_n/\Delta t$, where $\Delta\hat{\mathbf{z}}_n = \hat{\mathbf{z}}_n - \hat{\mathbf{z}}_{n-1}$ is the increment of the ISVs at the current step n. To obtain $\Delta\mathbf{z}_n$, alternative TCRNNs are proposed, as shown in Fig. 7.6, where the last two steps of the RNN infer the ISVs, $\hat{\mathbf{z}}_{n-1}$ and $\hat{\mathbf{z}}_n$, respectively. These TCRNN models enhance training efficiency.

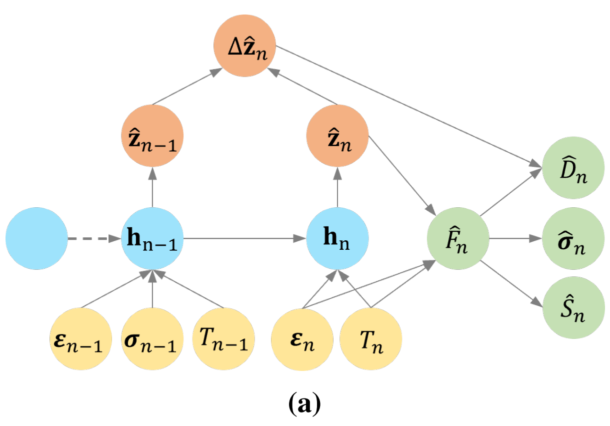
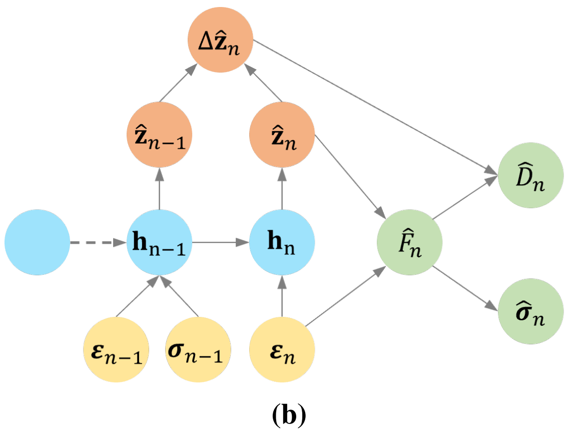

$\Delta\hat{\mathbf{z}}_n$
$\hat{\mathbf{z}}_{n-1}$
$\hat{\mathbf{z}}_n$
$\widehat{D}_n$
$\mathbf{h}_{n-1}$
$\mathbf{h}_n$
$\widehat{F}_n$
$\widehat{\boldsymbol{\sigma}}_n$
$\widehat{S}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\boldsymbol{\sigma}_{n-1}$
T_{n-1}
$\boldsymbol{\varepsilon}_n$
T_n
(a)

$\Delta\hat{\mathbf{z}}_n$
$\hat{\mathbf{z}}_{n-1}$
$\hat{\mathbf{z}}_n$
$\widehat{D}_n$
$\mathbf{h}_{n-1}$
$\mathbf{h}_n$
$\widehat{F}_n$
$\widehat{\boldsymbol{\sigma}}_n$
$\boldsymbol{\varepsilon}_{n-1}$
$\boldsymbol{\sigma}_{n-1}$
$\boldsymbol{\varepsilon}_n$
(b)

Figure 7.6. Computational graphs of (a) a thermodynamically consistent recurrent neural network (TCRNN) for non-isothermal processes and (b) a TCRNN for isothermal processes. The increment of the machine-extracted internal state, $\Delta\hat{\mathbf{z}}_n$, is computed and used for calculating the dissipation rate.

7.3.3 Model Training

The loss function is expressed as

$$Loss = \sum_n ||\bar{\sigma}_n - \hat{\bar{\sigma}}_n||_{L_1}^2 + \beta_1 ||\bar{F}_n - \hat{\bar{F}}_n||_{L_1}^2 + \beta_2 ||\bar{D}_n - \hat{\bar{D}}_n||_{L_1}^2 + \beta_3 ||\bar{S}_n - \hat{\bar{S}}_n||_{L_1}^2, \tag{7.15}$$

where β_i, $i = 1,..,3$, are regularization parameters. $||\cdot||_{L_1}$ denotes the L_1 norm, and β_3 is set to be zero if the data of the entropy is unavailable. The training consists of forward propagation and backward propagation. During the forward propagation, the machine-learned ISVs are implicitly embedded in the calculation of the predicted measurable quantities by following the thermodynamics principles. During the backward propagation, the errors of the measurable quantities are back-propagated to update the model's trainable parameters and refine machine-learned ISVs.

In some cases where the data of the dissipation rate D is unavailable, the non-negativity condition, i.e., $\hat{D} \geq 0$, can be imposed instead, which is resulted from the thermodynamics second law, Eq. (3.24a). To this end, the rectified linear unit (ReLU) can be utilized, i.e., ReLU(x) = $max(0,x) \geq 0$, which is positive only if x is positive. Hence, ReLU$(-\hat{D})$ is positive only if $\hat{D}$ is negative, which corresponds to violation of the non-negativity condition $\hat{D} \geq 0$. Including ReLU$(-\hat{D})$ to the loss function penalizes the violation of the non-negativity condition and enforces $\hat{D} \geq 0$ to be satisfied, which imposes a constraint on the network parameters during training. The loss function becomes

$$Loss = \sum_{n} ||\bar{\sigma}_n - \hat{\bar{\sigma}}_n||_{L_1}^2 + \beta_1 ||\bar{F}_n - \hat{\bar{F}}_n||_{L_1}^2 + \beta_2 ReLU(-\hat{D}_n). \tag{7.16}$$

Similarly, if the data of the Helmholtz free energy F is unavailable, the non-negativity condition, i.e., $\hat{F} \geq 0$, can be imposed by adding ReLU$(-\hat{F}_n)$ to the loss function,

$$Loss = \sum_{n} ||\bar{\sigma}_n - \hat{\bar{\sigma}}_n||_{L_1}^2 + \beta_1 ReLU(-\hat{F}_n) + \beta_2 ReLU(-\hat{D}_n). \tag{7.17}$$

In some cases where prior knowledge of certain ISVs are available, the TCRNN models can be trained in a *hybrid* mode by leveraging the existing ISVs and simultaneously inferring additional thermodynamically consistent ISVs that are essential to path-dependent behaviors. Considering $\mathbf{z}_n^p$ as the known ISVs and $\bar{\mathbf{z}}_n^p$ as the corresponding standardized quantity, the loss function becomes

$$Loss = \sum_{n} ||\bar{\sigma}_n - \hat{\bar{\sigma}}_n||_{L_1}^2 + \beta_1 ReLU(-\hat{F}_n) + \beta_2 ReLU(-\hat{D}_n) + \beta_4 ||\bar{\mathbf{z}}_n^p - \hat{\bar{\mathbf{z}}}_n^p||_{L_1}^2, \tag{7.18}$$

where the last term enables the TCRNN model to learn the existing ISVs. For the TCRNN model to infer additional essential ISVs, the prescribed internal state dimension, $|\hat{\mathbf{z}}|$, should be greater than the dimension of the existing ISVs, $|\mathbf{z}^p|$. Note that both existing and machine-learned ISVs are passed to the DNN to predict the Helmholtz free energy (Eq. 7.9) and the downstream calculations (Eqs. 7.13-7.14).

Apart from thermodynamics, the time (or self) consistency condition is critical for convergence of numerical approximation when $\Delta t \to 0$ [77].

$$\lim_{\Delta\varepsilon \to 0} \Delta\hat{\sigma} = 0. \tag{7.19}$$

To achieve the time consistency condition, the training set can be augmented by additional samples constituted by zero strain increment and zero stress increment at different material states (time steps), which enables the machine-learned material model to learn the time consistency condition from data. Alternatively, the self-consistency condition can be integrated into the RNN architecture by definition [93].

The optimal parameters of TCRNN are obtained by minimizing the loss functions (Eqs. (7.15)-(7.18)) using the open-source Pytorch library [214]. The trainable parameters are initialized from a uniform distribution $\mathscr{U}(-\sqrt{k}, \sqrt{k})$, where k is the reciprocal of the hidden dimension, $|\mathbf{h}|$. The Adam optimizer [215] is adopted for back-propagation training with an initial learning rate as 10^{-3}. To avoid over-fitting, a L_2-norm regularization on trainable parameters is imposed with a regularization parameter as 10^{-5}. During training, the model is evaluated by a validation set. When there is no improvement of the validation error after a prescribed number of epochs (set to be 100 in this study) the training terminates and the optimal model with a minimum validation error is retained, known as the *early stopping* procedure. Since the training dataset is standardized to balance feature contributions to the loss function, the training is less sensitive to the regularization parameters. Hence, the regularization parameters are set to be 1 unless further tuning is required to achieve better training performances.

7.4 Numerical Results

7.4.1 Modeling Elasto-Plastic Materials

To demonstrate the accuracy, robustness, and generalization performances, the proposed TCRNN is applied to model a material with synthetic data generated by the one-dimensional elasto-plastic material with kinematic hardening. The Helmholtz free energy potential is expressed as

$$F(\varepsilon,\varepsilon^p)=\frac{E}{2}(\varepsilon-\varepsilon^p)^2+\frac{H}{2}(\varepsilon^p)^2, \tag{7.20}$$

where $E = 100\ GPa$ is the Young's modulus; $H = 100\ GPa$ is the kinematic hardening parameter; ε is the total strain; ε^p is the plastic strain, which can be considered as a phenomenological ISV. The yield stress is $k = 100\ MPa$. The stress and the dissipation rate can be obtained by Eq. (3.31b-c) as follows

$$\sigma=\frac{\partial F}{\partial \varepsilon}=E(\varepsilon-\varepsilon^p), \tag{7.21a}$$

$$D=-\frac{\partial F}{\partial \varepsilon^p}\dot{\varepsilon}^p=(\sigma-H\varepsilon^p)\dot{\varepsilon}^p. \tag{7.21b}$$

The dataset is generated by Eqs. (7.20)-(7.21), which contains five samples with the same stress-strain path (two loading-unloading cycles), as show in Fig. 7.7. The only difference in these samples is the strain increment, ranging from 3.75×10^{-5} to 7.5×10^{-5}. The data of the sample with a strain increment of 5.0×10^{-5} is used to train the TCRNN. The remaining samples are in the testing set to evaluate the trained model.

To address the issue of error propagation in the test mode due to the teacher forcing training, as discussed in Section 7.2.2, and enhance model accuracy and robustness, *stochasticity* is introduced to the stress data so that the model learns the uncertainties of the input conditions resembling those in the test mode. Random perturbations are generated from a normal (Gaussian) distribution with a zero mean and a standard deviation of $r\times\sigma_{max}$, where σ_{max} is the maximum stress in the data and r is a user-defined parameter to control the level of randomness. Fig. 7.7 shows the original stress-strain data in a black solid line and the randomly perturbed data in red circles. During supervised training, the noiseless stress is the ground truth and the input stress variable is no longer deterministic but rather stochastic, contributed by the random perturbations.

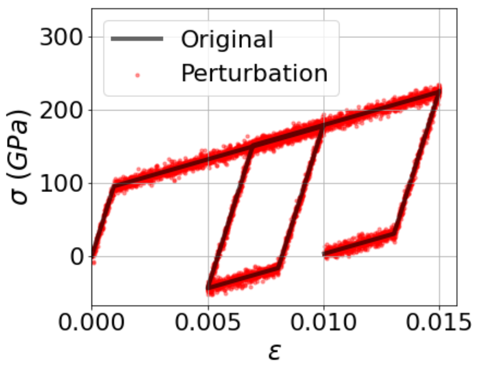


Figure 7.7. Training data with random perturbations to enhance prediction accuracy and robustness of TCRNNs, where the black solid line denotes the original data and red circles denote randomly perturbed data. $r = 0.3$ is employed to generate the random perturbations.

The TCRNN model based on the time rates of ISVs ($\dot{\hat{\mathbf{z}}}$), as shown in Fig. 7.4(b), is employed in this example. A GRU is used to infer the ISV $\hat{\mathbf{z}}$ and describe its evolution by following the thermodynamics second law. The GRU consists of one hidden layer for all affine transformations in Eq. (7.2) with the model complexity represented by the dimension of the hidden state, $|\mathbf{h}|$. The DNN trained to predict the Helmholtz free energy contains one hidden layer with the number of neurons the same as $|\mathbf{h}|$. Since the data of the free energy F and the dissipation rate D can be obtained by Eq. (7.21) in this example, the loss function in Eq. (7.15) is employed with $\beta_1 = \beta_2 = 1$ and $\beta_3 = 0$. A relative error used to measure the prediction accuracy is defined as follows

$$e = \frac{||\Sigma - \hat{\Sigma}||_{L_2}}{||\Sigma||_{L_2}}, \tag{7.22}$$

where Σ and $\hat{\Sigma}$ contain the stress data and stress predictions at all time steps of a loading path, respectively.

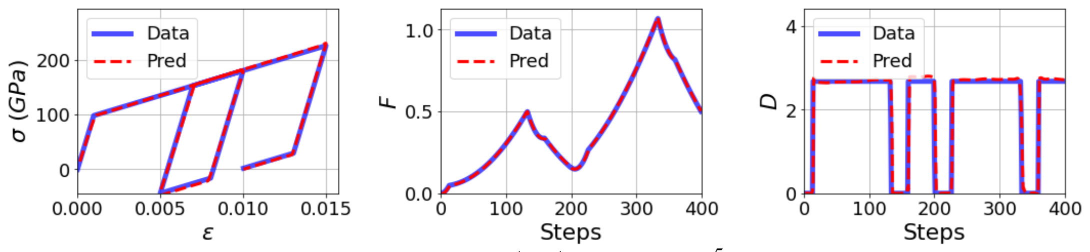
In the following subsections, the effects of the strain increments on model performance are first investigated. Since the machine-inferred ISVs are critical to the accuracy of path-dependent materials modeling, various factors that can affect the quality of the machine-inferred ISVs are investigated, including the number of RNN steps, the internal state dimension ($|\hat{\mathbf{z}}|$), and the model complexity ($|\mathbf{h}|$). Further, the generalization capabilities of the TCRNN model are examined.

Effects of Strain Increments

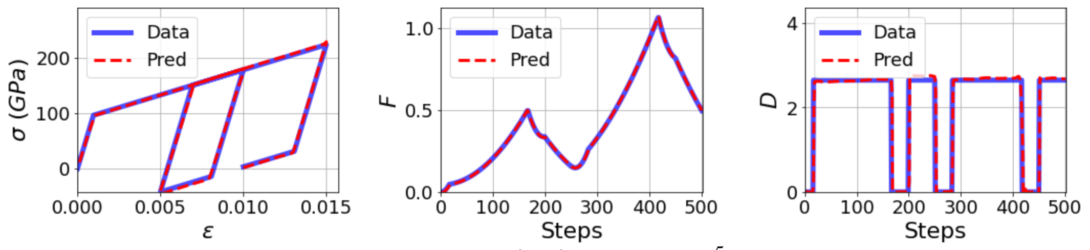
We first investigate how the strain increment affects the prediction accuracy of the TCRNN model. The model has a scalar ISV and a hidden state dimension of 30 ($|\mathbf{h}| = 30$). Fig. 7.8 compares the predictions with data, where the case with a green color line is used for training and those with blue color lines are used for testing. The trained model achieves 1.1% relative error (Eq. (7.22)) on the stress prediction for the training case and 1.9% mean relative error for the testing cases. The mean relative error is obtained by averaging the relative errors (Eq. (7.22)) of all cases. The good agreement between predictions and data for all quantities, including the stress, the Helmholtz free energy, and the dissipation rate demonstrates that the model maintains high prediction accuracy and robustness as the strain increment varies. Fig. 7.9 shows that the machine-learned ISV is monotonically correlated with the phenomenological ISV, which demonstrates the capability of the TCRNN model in extracting mechanistically and thermodynamically consistent ISV essential to dissipative elasto-plastic material behaviors.

Effects of The Number of RNN Steps

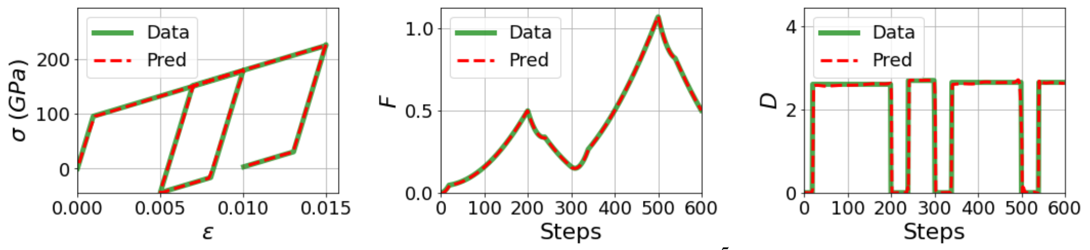
In the second example, TCRNN models with a scalar ISV and various RNN steps (including history and current steps) are examined. Note that the model complexity is independent of the number of RNN steps due to parameter sharing across all RNN steps. For a relatively compact model with $|\mathbf{h}| = 20$, Fig. 7.10(a) shows that the relative errors of training and testing samples decrease significantly when the number of RNN steps reaches 4, a critical number of RNN steps, which is expected since increasing the number of RNN steps enables the model to



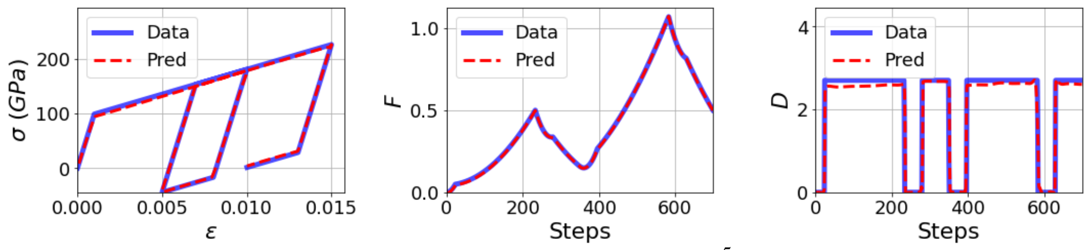
(a) Testing: $|\Delta\varepsilon| = 7.5 \times 10^{-5}$



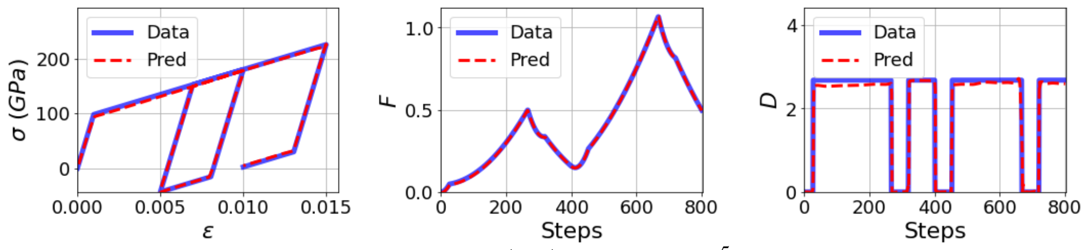
(b) Testing: $|\Delta\varepsilon| = 6.0 \times 10^{-5}$



(c) Training: $|\Delta\varepsilon| = 5.0 \times 10^{-5}$



(d) Testing: $|\Delta\varepsilon| = 4.29 \times 10^{-5}$



(e) Testing: $|\Delta\varepsilon| = 3.75 \times 10^{-5}$

Figure 7.8. Comparison of predictions of the TCRNN with data: (a) testing case with a strain increment of $|\Delta\varepsilon| = 7.5 \times 10^{-5}$; (b) testing case with a strain size of $|\Delta\varepsilon| = 6.0 \times 10^{-5}$; (c) training case with a strain size of $|\Delta\varepsilon| = 7.5 \times 10^{-5}$; (d) testing case with a strain size of $|\Delta\varepsilon| = 4.29 \times 10^{-5}$; (e) testing case with a strain size of $|\Delta\varepsilon| = 3.75 \times 10^{-5}$.

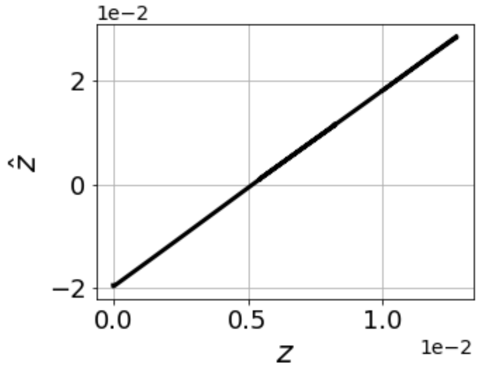


Figure 7.9. Correlation between the machine-learned internal state variable and the phenomenological internal state variable.

extract more accurate path-dependent features from longer-term stress-strain history. As the number RNN steps further increases, the relative errors of training and testing samples remain at a plateau, with around 0.4% and 1.6% relative errors, respectively. The plateau indicates that further increasing the number of RNN steps does not improve the quality of the machine-inferred ISVs and thus the model accuracy, which could be potentially limited by $|\hat{\mathbf{z}}|$ or $|\mathbf{h}|$. For a more complex model with $|\mathbf{h}| = 50$, Fig. 7.10(b) shows a similar convergence behavior but the critical number of RNN steps increases to 5. This shows that the number of RNN steps play an important role in model accuracy and performance. Unnecessarily increasing the model complexity may lead to an increase in the number of RNN steps for achieving the same level of accuracy.

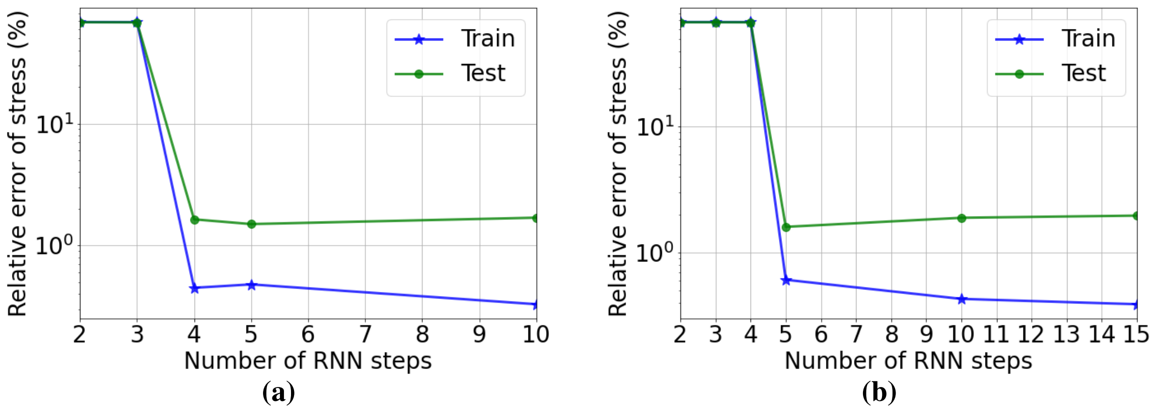


Figure 7.10. Effects of the number of RNN steps on the accuracy of models with a scalar internal state variable and: (a) a hidden state dimension $|\mathbf{h}| = 20$; (b) a hidden state dimension $|\mathbf{h}| = 50$.

Effects of Internal State Dimension

The internal state dimension ($|\hat{\mathbf{z}}|$) has a direct impact on the quality of the machine-inferred ISVs and thus the model performance. If $|\hat{\mathbf{z}}|$ is too small, the TCRNN model cannot capture all important thermodynamically consistent path-dependent features even if the number of RNN steps and model complexity are sufficient. In this example, TCRNN models with 5 RNN steps, $|\mathbf{h}| = 20$ and various internal state dimensions are examined. Fig. 7.11(a) shows that as the dimension of the ISV increases from 1 to 5, the relative errors of training and testing samples remain at a plateau, with around 0.5% and 1.5% relative errors, respectively. It indicates that a scalar ISV is sufficient for effectively capturing the path-dependent material behavior in this case. The convergence of the model performance against the internal state dimension is particularly important. In practice, the internal state dimension of path-dependent materials is often unknown a priori. The convergence property shows that the TCRNN model remains accurate and robust even if an excessive internal state dimension is prescribed. This convergence property also allows one to identify the optimal $|\hat{\mathbf{z}}|$ given measurable material states of path-dependent materials.

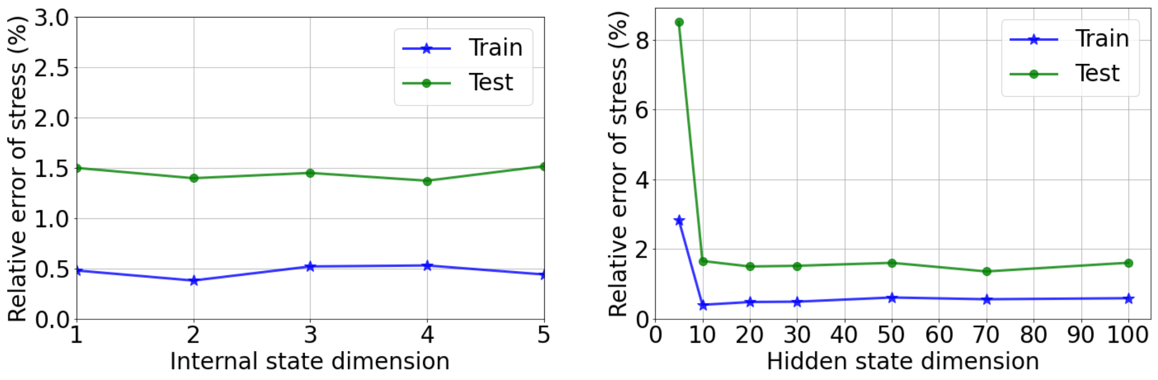


Figure 7.11. (a) Effects of the internal state dimension on the accuracy of models with 5 RNN steps and a hidden state dimension $|\mathbf{h}| = 20$. (b) Effects of the network complexity (hidden state dimension) on the accuracy of models with a scalar internal state variable and 5 RNN steps.

Effects of Model Complexity

The model complexity represented by the hidden state dimension is another important factor influencing the quality of the machine-inferred ISVs and model performance since the ISVs are inferred from the hidden states that directly capture the essential stress-strain path-dependent features. If the hidden state dimension is too small, important stress-strain path-dependent features could be lost, leading to inaccurate machine-inferred ISVs and poor model performance. In this example, the effects the TCRNN model complexity (hidden state dimension) are investigated. The TCRNN models examined have a scalar ISV, 5 RNN steps, various hidden state dimensions ranging from 5 to 100. Fig. 7.11(b) shows that as the hidden state dimension increases, the relative errors of training and testing samples decrease and then reach a plateau, with around 0.5% and 1.5% relative errors, respectively. This shows that a compact network is sufficient to achieve a satisfactory accuracy in this example and increasing the model complexity does not improve the model accuracy.

Model Generalization

In the following examples, three variables are considered to evaluate the generalization performances of the TCRNN model, including the loading strain per cycle, the unloading strain per cycle, and the number of (loading-unloading) cycles. The TCRNN model with 15 RNN steps, a scalar ISV, and $|\mathbf{h}| = 30$ is employed.

In the first test, we consider a two-dimensional parameter space constituted by the loading strain per cycle and the unloading strain per cycle. The dataset contains 16 cases with the same number of loading-unloading cycles but with different loading and unloading strains per cycle. Fig. 7.12 shows the comparison between the predicted stress and the data, where case 1, 4, 9, and 16, located at the corners in the figure, are used for training with the data marked with the green solid lines, and the remaining cases are used for testing with the data marked with the blue solid lines. From top to bottom, the loading strain per cycle increases from 10^{-2} to 1.4×10^{-2}. From left to right, the unloading strain per cycle increases from 5×10^{-3} to 5.5×10^{-3}. The mean relative errors of the training and testing cases are 3.1% and 2.8%, respectively. The good agreement between the data and the predictions demonstrates that the trained TCRNN model can successfully predict the testing cases within the prescribed parameter space.

In the second test, we consider a two-dimensional parameter space constituted by the number of loading-unloading cycles and the loading strain per cycle. The dataset contains 16 cases with the same unloading strain per cycle, 5.5×10^{-3}. Fig. 7.13 shows the comparison between the predicted stress and the data, where case 1, 4, 9, and 16, located at the corners in the figure, are used for training with the data marked with the green solid lines, and the remaining cases are used for testing with the data marked with the blue solid lines. From top to bottom, the number of loading-unloading cycles increases from 1 to 4. From left to right, the loading strain per cycle increases from 10^{-2} to 1.4×10^{-2}. The mean relative errors of the training and testing cases are 2.3% and 3.4%, respectively. The good agreement between the data and the predictions further demonstrates the strong generalization ability of the TCRNN constitutive model.

7.4.2 Modeling Soil under Cyclic Shear Loading

The effectiveness of the proposed TCRNN constitutive model is further evaluated by modeling undrained soil (sand) under cyclic shear loading [216, 217]. The experimental data is collected from the undrained soil samples under initial triaxial confinement of $40kPa$ and cyclic shear loading. A cyclic stress ratio (CSR) is defined as the ratio of the maximum shear stress to the initial vertical stress. The experimental data contains the shear strain, the vertical strain, the shear stress, and the vertical stress. Fig. 7.14 shows the experimental data with a CSR of 0.15, 0.16, and 0.17, measured with a constant time step size as 8×10^{-3}. The stress-strain relationships are highly nonlinear and path-dependent due to coupling effects of changes in volume, matric suction, degree of saturation, effective stress, shear modulus, etc. [218]. Modeling such path-dependent material behaviors by a phenomenological approach is challenging and complicated, which often relies on certain phenomenological ISVs.

Given only the stress-strain data, data-driven models and phenomenological models that

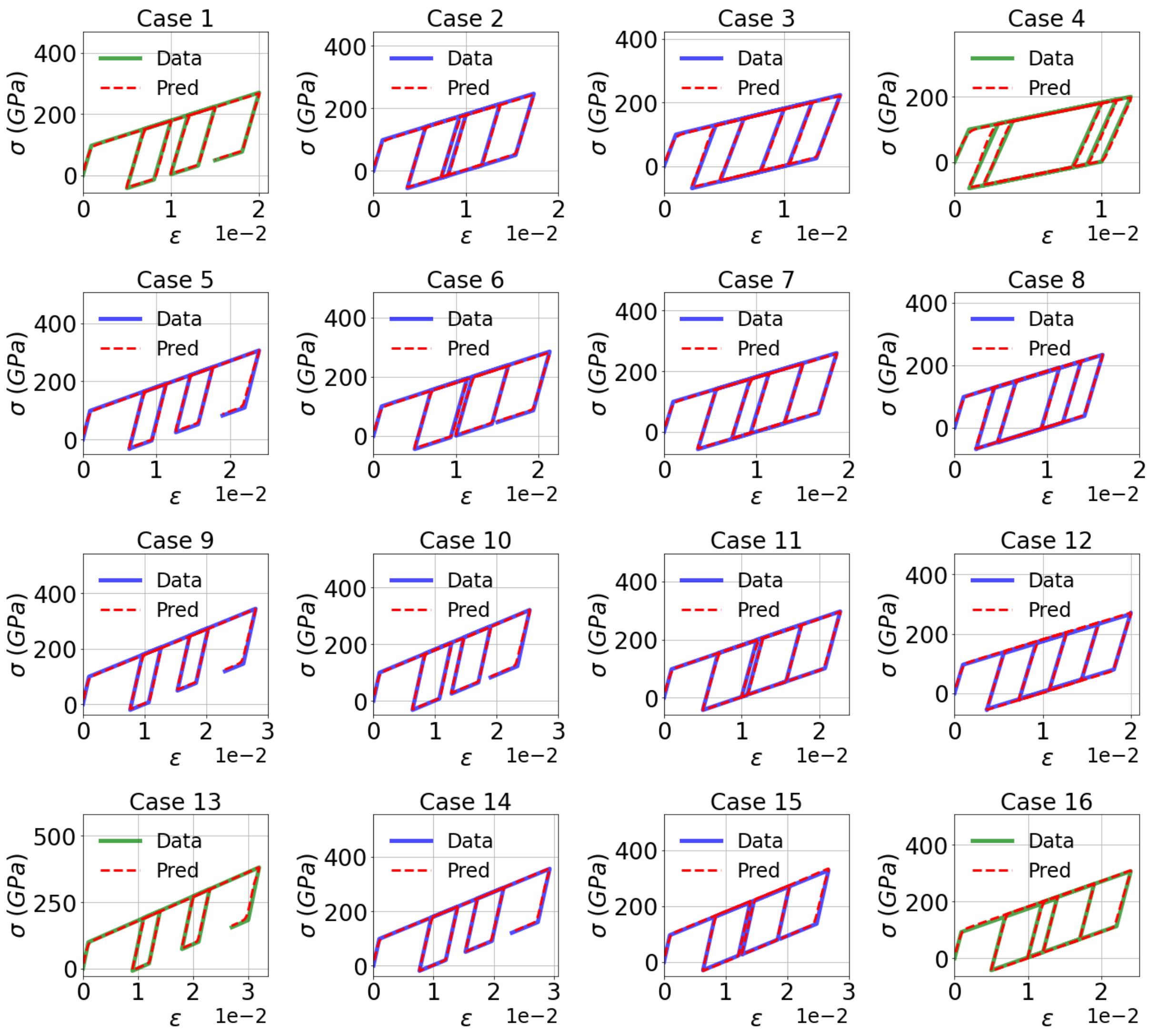


Figure 7.12. Comparison between the predicted stress of the TCRNN model and the data. The parameter space is constituted by the loading strain per cycle and the unloading strain per cycle. From top to bottom, the loading strain per cycle increases from 10^{-2} to 1.4×10^{-2}. From left to right, the unloading strain per cycle increases from 5×10^{-3} to 5.5×10^{-3}. the The training data are denoted by the green color lines whereas the testing cases are denoted by the blue color lines. The predictions are denoted by the red dash lines.

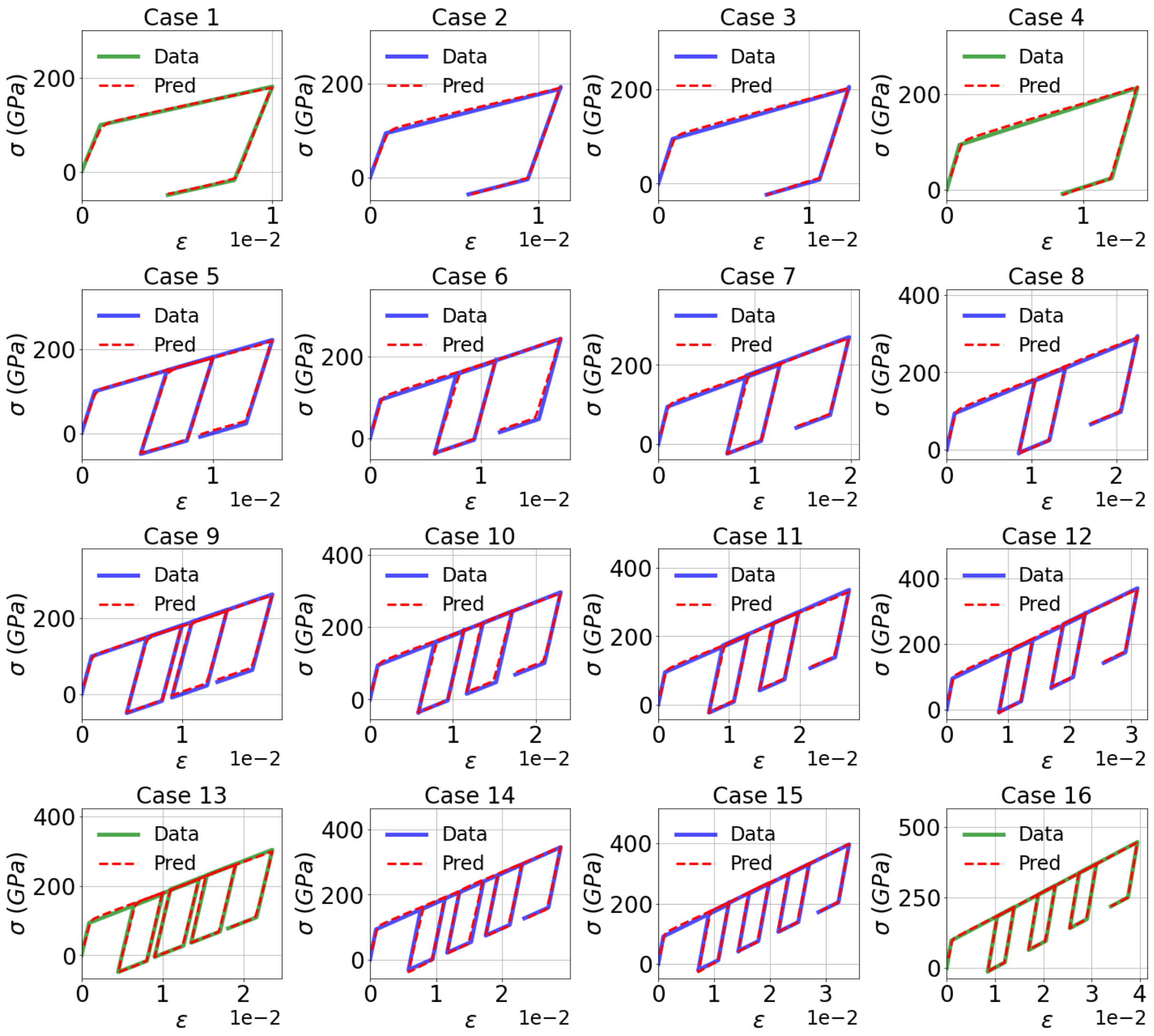
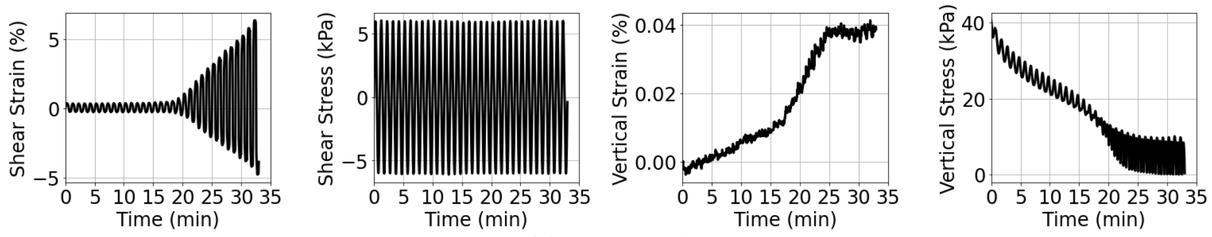
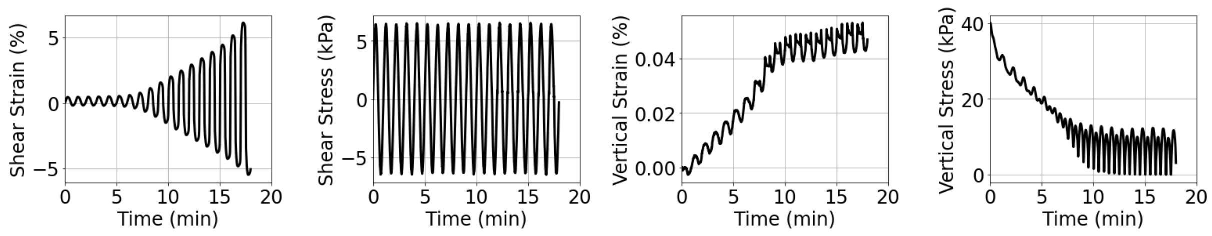


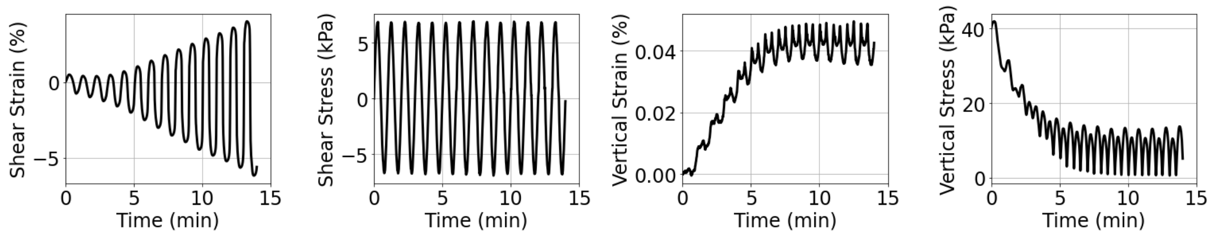
Figure 7.13. Comparison between the predicted stress of the TCRNN model and the data. The parameter space is constituted by the loading strain per cycle and the number of loading-unloading cycles. From top to bottom, the number of loading-unloading cycles increases from 1 to 4. From left to right, the loading strain per cycle increases from 10^{-2} to 1.4×10^{-2}. The training data are denoted by the green color lines whereas the testing cases are denoted by the blue color lines. The predictions are denoted by the red dash lines.



(a) CSR=0.15



(b) CSR=0.16



(c) CSR=0.17

Figure 7.14. Experimental data of undrained soil under cyclic simple shear loading with: (a) CSR=0.15; (b) CSR=0.16; (c) CSR=0.17.

require pre-defined ISVs cannot be applied. In contrast, the proposed TCRNN model can be effectively applied since it only requires measurable material states, and the model is capable of inferring essential ISVs from the measurable material states by following the thermodynamics principles.

The TCRNN based on the increment of ISVs ($\Delta\hat{\mathbf{z}}$), as shown in Fig. 7.6(b), is employed in this example. A GRU is used to infer the ISV $\hat{\mathbf{z}}$ in Eq. (7.8) and describe its evolution by following the thermodynamics second law in Eq. (3.22). Note that although the time step size of the experimental data is constant in this example, it is not required by the TCRNN to infer the incremental ISVs $\Delta\hat{\mathbf{z}}$. The GRU consists of one hidden layer for all affine transformations in Eq. (7.2) with the model complexity represented by the hidden state dimension, $|\mathbf{h}|$. The DNN trained to predict the Helmholtz free energy contains one hidden layer with the number of neurons the same as $|\mathbf{h}|$. Since the training data contains only stresses and strains, the loss function in Eq. (7.17) is employed with $\beta_1 = \beta_2 = 1$. The experimental data with a CSR of 0.15 and 0.17 are used for training, while the experimental data with a CSR of 0.16 is used for testing. The effects of the number of RNN steps, the internal state dimension, and the model complexity on the model performance are investigated.

Effects of The Number of RNN Steps

Given TCRNN models with an internal state dimension ($|\hat{\mathbf{z}}|$) of 2 and a hidden state dimension ($|\mathbf{h}|$) of 30, the number of RNN steps is varied from 5 to 60 and its influences on the model prediction accuracy are shown in Fig. 7.15(a) As the number of RNN steps increases, the relative errors of training and testing samples decrease and eventually converge to a plateau, with values around 3% and 11%, respectively. The plateau indicates that further increasing the number of RNN steps does not improve the model accuracy.

Effects of Internal State Dimension

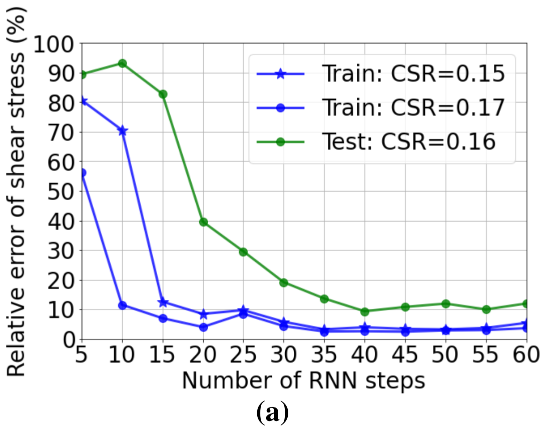
The internal state dimension ($|\hat{\mathbf{z}}|$) required to effectively model the path-dependent material behaviors is unknown a priori, which depends on the complexity of the path-dependent behaviors. Here, we investigate the effects of $|\hat{\mathbf{z}}|$ on model prediction accuracy, which is varied from 1 to 10, while the number of RNN steps and $|\mathbf{h}|$ are fixed as 40 and 30, respectively. Fig. 7.15(b) shows that the relative errors of training and testing samples are large when the machine-inferred ISV is a scalar, indicating that a scalar ISV is insufficient to capture all essential path-dependent features. As $|\hat{\mathbf{z}}|$ increases, the relative errors of training and testing samples decrease and then reach a plateau, with values around 2.7% and 12%, respectively, which shows that the TCRNN model remains accurate and robust even if an excessive $|\hat{\mathbf{z}}|$ is prescribed. The convergence behavior also allows one to identify the optimal $|\hat{\mathbf{z}}|$, around 2 in this example, given the measurable material states of path-dependent materials.

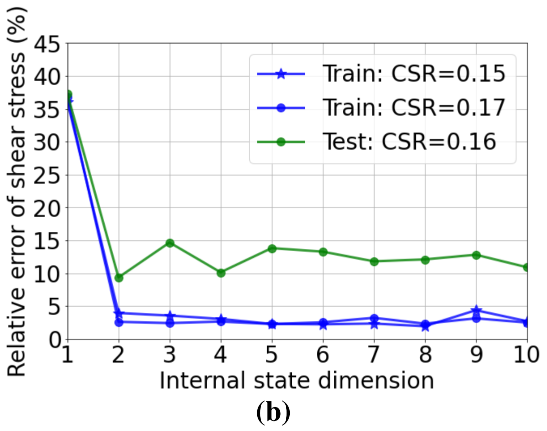
Effects of Model Complexity

Lastly, $|\hat{\mathbf{z}}|$ and the number of RNN steps are fixed as 5 and 40, respectively, while $|\mathbf{h}|$ is varied from 5 to 100 to investigate the effects of model complexity ($|\mathbf{h}|$) on model performance. Fig. 7.15(c) shows that the relative errors of training and testing samples decrease as $|\mathbf{h}|$ increases and eventually reach a plateau, with values around 2.8% and 14%, respectively. The plateaus in the convergence curves indicate that further increasing the model complexity does not improve the model accuracy.

Model Generalization

Fig. 7.16 compares shear stress experimental data with the predictions of the trained TCRNN model that employs 40 RNN steps, $|\hat{\mathbf{z}}| = 2$, and $|\mathbf{h}| = 30$. The relative errors of training and testing samples are around 3.2% and 9.4%, respectively. It shows that the TCRNN model is able to learn the path-dependent material behaviors from the measurable material states under given loading conditions and effectively predicts the path-dependent responses under





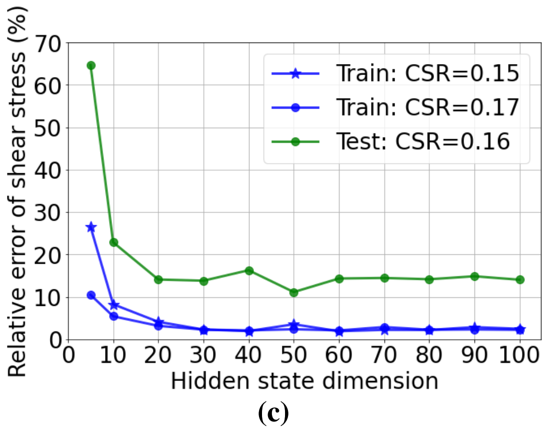
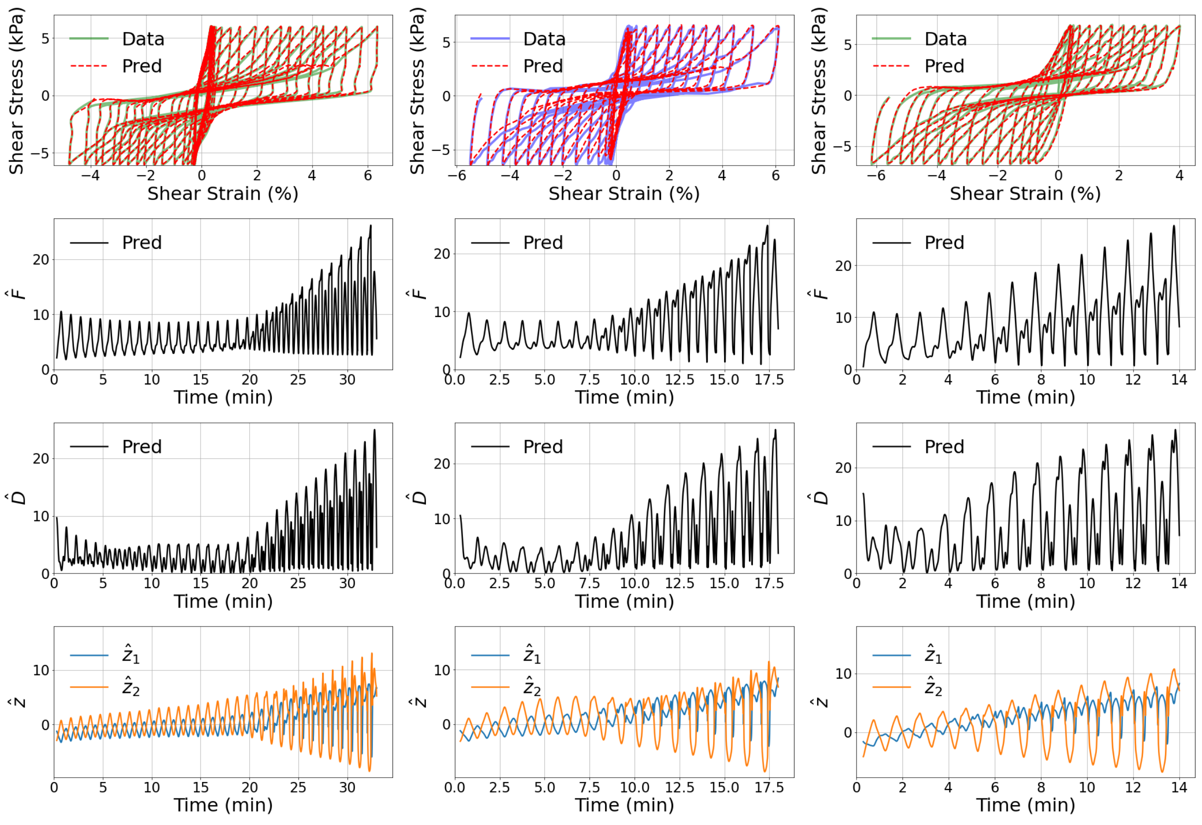


Figure 7.15. Effects of various parameters on model accuracy: (a) the number of RNN steps (with an internal state dimension $|\hat{\mathbf{z}}| = 2$ and a hidden state dimension $|\mathbf{h}| = 30$); (b) the internal state dimension (with 40 RNN steps and a hidden state dimension $|\mathbf{h}| = 30$); (c) the model complexity (with an internal state dimension $|\hat{\mathbf{z}}| = 5$ and 40 RNN steps).

untrained loading conditions, further demonstrating the generalization ability and effectiveness of the TCRNN model in practical applications. Further, the trained TCRNN material model is thermodynamically consistent, which is verified by the non-negative predicted free energy and the predicted dissipation rates that satisfy the thermodynamics second law, as shown in the second and the third rows of Fig. 7.16, respectively. The histories of the machine-learned ISVs are shown in the last row of Fig. 7.16, revealing interesting path-dependent patterns similar to the behaviors of the predicted free energy and dissipation rate.



(a) Training: CSR=0.15 **(b)** Testing: CSR=0.16 **(c)** Training: CSR=0.17

Figure 7.16. Comparison of predictions of the TCRNN with data: (a) the training case with a CSR=0.15; (b) the testing case with a CSR=0.16; (c) the training case with a CSR=0.17. The first row compares shear stress-strain relationships. The second row shows the predicted Helmholtz free energy. The third row shows the predicted dissipation rate. The last row shows the machine-learned internal state variables. The TCRNN model employed has 40 RNN steps, an internal state dimension $|\hat{\mathbf{z}}| = 2$, and a hidden state dimension $|\mathbf{h}| = 30$.

7.5 Summary

In this study, we introduced a machine-learned internal state variable (ISV) approach for data-driven modeling of path-dependent materials, which is thermodynamically consistent and relies purely on the measurable material states. The proposed TCRNN constitutive models consist of two main components: an RNN that infers ISVs (Eq. (7.8)) and describes their evolution by following the thermodynamics second law (Eq. (3.22)), and a DNN that predicts the Helmholtz free energy (Eq. (7.9)) given strain, ISVs, and temperature (for non-isothermal processes). Two TCRNN constitutive models are developed, one based on the time rates of ISVs ($\dot{\hat{\mathbf{z}}}$), as shown in Fig. 7.4, and the other one based on the increments of ISVs ($\Delta\hat{\mathbf{z}}$), as shown in Fig. 7.6. The latter model shows an enhanced efficiency as it utilizes an approximation of $\dot{\hat{\mathbf{z}}}$ for the calculation of dissipation rate and avoids time-consuming differentiation of the RNN outputs with respect to all RNN inputs. Model robustness and accuracy is enhanced by introducing *stochasticity* to the training data to account for uncertainties of input conditions in the testing.

In the demonstration of modeling elasto-plastic materials, the parametric study shows that the model accuracy converges as the number of RNN steps, the internal state dimension, and the model complexity increase. All these factors play an important role in the model performance. Given path-dependent material behaviors, there exists an optimal internal state dimension to capture the essential path-dependent features by the TCRNN model. It has been shown that the TCRNN model remains accurate and robust even if an excessive internal state dimension is prescribed. The monotonic correlation between the machine-inferred and the phenomenological ISV of the elasto-plastic material demonstrates that the TCRNN constitutive model can infer mechanistically and thermodynamically consistent ISVs. The proposed TCRNN constitutive model is shown to remain robust against various strain increments and have strong generalization capabilities.

The effectiveness of the proposed TCRNN constitutive model is further demonstrated by modeling undrained soil under cyclic shear loading using experimental data, where only measurable material states (stresses and strains) are available. A similar convergence behaviors of the model accuracy are observed from a parametric study of the number of RNN steps, the internal state dimension, and the model complexity. The generalization capability of the TCRNN constitutive model is demonstrated by the effective prediction of the thermodynamically consistent response of undrained soil under the loading conditions different from the ones used in training, which reveals the promising potential of the proposed method to model complex path-dependent materials behaviors in real applications.

The proposed TCRNN constitutive model is general and applicable to model a wide range of path-dependent materials. It is efficient and can be applied to accelerate large-scale multi-scale simulations with complex microstructures and path-dependent material systems. To investigate reliability of model predictions, a future extension would be to integrate uncertainty quantification [32] into the proposed TCRNN model.

7.6 Acknowledgement

This chapter, in part, is a reprint of the material as it appears in: “Xiaolong He and Jiun-Shyan Chen. Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, page 115348, 2022”. The dissertation author is the primary investigator and author of this paper.

Chapter 8

gLaSDI: Parametric Physics-informed Greedy Latent Space Dynamics Identification

8.1 Introduction

Physical simulations have played an increasingly significant role in developments of engineering, science, and technology. The widespread applications of physical simulations in digital twins systems [107, 108] is one recent example. Many physical processes are mathematically modeled by time-dependent nonlinear partial differential equations (PDEs). As it is difficult or even impossible to obtain analytical solutions for many highly complicated problems, various numerical methods have been developed to approximate the analytical solutions. However, due to the complexity and the domain size of problems, high-fidelity forward physical simulations can be computationally intractable even with high performance computing, which prohibits their applications to problems that require a large number of forward simulations, such as design optimization [29, 30], optimal control [31], uncertainty quantification [32, 33], and inverse analysis [33, 34].

In recent years, several reduced-order model (ROM) methods have been integrated with latent-space learning algorithms. Kim, et al. [154] proposed a DeepFluids framework in which the autoencoder was applied for nonlinear projection and a latent-space time integrator was used to approximate the evolution of the solutions in the latent space. Xie, et al. [57] applied the POD for linear projection and a multi-step NN to propagate the latent-space dynamical solutions. Hoang, et al. [155] applied the POD to compress space-time solution space to obtain space-time reduced-order basis and examined several surrogate models to map input parameters to space-time basis coefficients, including multivariate polynomial regression, *k*-nearest neighbors (KNNs), random forest, and NNs. Kadeethum, et al. [61] compared performance of the POD and autoencoder compression along with various latent space interpolation techniques, such as radial basis function and artificial neural networks. However, the latent-space dynamics models of these methods are complex and lack interpretability.

To improve the interpretability and generalization capability, many non-intrusive ROMs have been developed to explicitly identify interpretable governing laws of latent-space dynamics by operator inference [158–160, 166–175] and parametric models, such as the sparse identification of nonlinear dynamics (SINDy) [52, 58, 165]. For example, Qian, et al. [159] introduced a lifting map to transform non-polynomial physical dynamics to quadratic polynomial dynamics and then combined POD-based linear projection with operator inference to identify quadratic reduced models for dynamical systems. Bai and Peng [58] proposed parametric non-intrusive ROMs that combine the POD for linear projection with regression surrogates to approximate dynamical systems of latent variables, including support vector machines with kernel functions, tree-based methods, KNNs, vectorial kernel orthogonal greedy algorithm (VKOGA), and SINDy. The ROMs integrated with VKOGA and SINDy deliver superior cost versus error trade-off. Due to the limitation of the POD-based linear projection, these non-intrusive ROMs have difficulties with advection-dominated problems. To address this challenge, Issan and Kramer [176] recently proposed a non-intrusive ROM based on shifted operator inference by transforming the original coordinate frame of dynamical systems to a moving coordinate frame in which the dynamics are absent of translation and rotation. Fries, et al. [9] proposed a parametric latent space dynamics identification (LaSDI) framework as a generalization of aforementioned non-intrusive ROMs built upon latent-space dynamics identification, since it allows linear or nonlinear projection and enables latent-space dynamics to be captured by flexible dynamics identification (DI) models based on general nonlinear functions. In this study [9], POD-based linear projection is compared with autoencoder-based nonlinear projection and it is demonstrated that local latent-space dynamics can be leveraged to enhance predictivity of the ROM, especially when the dynamics is highly complex and localized. However, a sequential training procedure was adopted for the autoencoder and the DI models, the lack of interaction between the autoencoder and the DI models leads to strong dependency of the complexity and quality of the latent-space dynamics on the autoencoder architecture, which could pose challenges to the subsequent training of the DI models and thus affect the model performances. Most importantly, all the above-mentioned approaches rely on predefined training samples, such as uniform or Latin hypercube sampling that may not be optimal in terms of the number of samples for achieving the best model performance in the prescribed parameter space. As the generation of the simulation data can be computationally expensive, it is important to minimize the number of samples.

In this study, we propose a parametric adaptive greedy latent space dynamics identification (gLaSDI) framework for accurate, efficient, and robust physics-informed data-driven reduced-order modeling. To maximize and accelerate the exploration of the parameter space for optimal performance, an adaptive greedy sampling algorithm integrated with a physics-informed residual-based error indicator and random-subset evaluation is introduced to search for the optimal and minimal training samples on the fly. The proposed gLaSDI framework contains an autoencoder for nonlinear projection to discover intrinsic latent representations and a set of local DI models to capture local latent-space dynamics, which is further exploited by an efficient KNN convex interpolation scheme. The autoencoder training and dynamics identification in the gLaSDI take place interactively to achieve an optimal identification of simple latent-space dynamics.

The remainder of this chapter is organized as follows. The governing equations of dynamical systems is introduced in Section 8.2. In Section 8.3, the ingredients of the proposed gLaSDI framework, the mathematical formulations, the training and testing algorithms are introduced. In Section 8.4, the effectiveness and capability of the proposed gLaSDI framework are examined by modeling various nonlinear dynamical problems, including Burgers equations, nonlinear heat conduction, and radial advection. The effects of various factors on model performance are investigated, including the number of nearest neighbors for convex interpolation, the latent-space dimension, the complexity of the DI models, and the size of the parameter space. A performance comparison between uniform sampling, i.e., LaSDI, and the physics-informed greedy sampling, i.e., gLaSDI, is also presented. Concluding remarks and discussions are summarized in Section 8.5.

8.2 Governing equations of dynamical systems

A parameterized dynamical system characterized by a system of ordinary differential equations (ODEs) is considered

$$\frac{d\mathbf{u}(t;\mu)}{dt} = \mathbf{f}(\mathbf{u},t;\mu), \quad t \in [0,T], \tag{8.1a}$$

$$\mathbf{u}(0;\mu) = \mathbf{u}_0(\mu) \tag{8.1b}$$

where $T \in \mathbb{R}_+$ is the final time; $\mu \in \mathscr{D} \subseteq \mathbb{R}^{n_\mu}$ is the parameter in a parameter domain $\mathscr{D}$. $\mathbf{u}(t;\mu) : [0,T] \times \mathscr{D} \to \mathbb{R}^{N_u}$ is the parameterized time-dependent solution to the dynamical system; $\mathbf{f} : \mathbb{R}^{N_u} \times [0,T] \times \mathscr{D} \to \mathbb{R}^{N_u}$ denotes the velocity of $\mathbf{u}$; $\mathbf{u}_0$ is the initial state of $\mathbf{u}$. Eq. (8.1) can be considered as a semi-discretized equation of a system of partial differential equations (PDEs) with a spatial domain $\Omega \subseteq \mathbb{R}^d, d \in \mathbb{N}(3)$, and $\mathbb{N}(N) := \{1,...,N\}$. Spatial discretization can be performed by numerical methods, such as the finite element method.

A uniform time discretization is considered in this study, with a time step size $\Delta t \in \mathbb{R}_+$ and $t_n = t_{n-1} + \Delta t$ for $n \in \mathbb{N}(N_t)$ where $t_0 = 0$, $N_t \in \mathbb{N}$. Various explicit or implicit time integration schemes can be applied to solve Eq. (8.1). For example, with the implicit backward Euler time integrator, the solutions to Eq. (8.1) can be obtained by solving the following nonlinear system of equations

$$\mathbf{u}_n = \mathbf{u}_{n-1} + \Delta t \mathbf{f}_n, \tag{8.2}$$

where $\mathbf{u}_n := \mathbf{u}(t^n;\mu)$, and $\mathbf{f}_n := \mathbf{f}(\mathbf{u}(t^n;\mu),t^n;\mu)$. The residual function of Eq. (8.2) is expressed as

$$\mathbf{r}(\mathbf{u}_n;\mathbf{u}_{n-1},\mu) = \mathbf{u}_n - \mathbf{u}_{n-1} - \Delta t \mathbf{f}_n. \tag{8.3}$$

Solving the dynamical system of equation (Eq. (8.1)) could be computationally expensive, especially when the solution dimension (N_u) is large and the computational domain (Ω) is geometrically complex. In this work, an efficient and accurate data-driven reduced-order modeling framework based on physics-informed greedy latent space dynamics identification is proposed, which will be discussed in details in the next section.

8.3 gLaSDI

The ingredients of the proposed physics-informed parametric adaptive greedy latent space dynamics identification (gLaSDI) framework are introduced in this section, including autoencoders, dynamics identification (DI) models, *k*-nearest neighbors (KNN) convex interpolation, and an adaptive greedy sampling procedure with a physics-informed error indicator. An autoencoder is trained to discover an intrinsic nonlinear latent representation of high-dimensional data from dynamical PDEs, while training cases are sampled on the fly and associated local DI models are trained simultaneously to capture localized latent-space dynamics. An interactive training procedure is employed for the autoencoder and local DI models, which is referred to as the *interactive Auto-DI training*, as shown in Fig. 8.1. The interaction between the autoencoder and local DI models enables identification of simple and smooth latent-space dynamics and therefore accurate and efficient data-driven reduced-order modeling.

The physics-informed adaptive greedy sampling can be performed on either a continuous parameter space ($\mathscr{D}$) or a discrete parameter space ($\mathscr{D}^h \subseteq \mathscr{D}$). In the following demonstration, a discrete parameter space ($\mathscr{D}^h$) is considered. $\mathbb{D} \subseteq \mathscr{D}^h$ denotes a set of N_μ selected training sample points.

Let us consider a training sample point $\mu^{(i)} \in \mathscr{D}^h$, $i \in \mathbb{N}(N_\mu)$ and $\mathbf{u}_n^{(i)} \in \mathbb{R}^{N_u}$ as the solution at the n-th time step of the dynamical system in Eq. (8.1) with the training sample point $\mu^{(i)}$. The solutions at all time steps are arranged in a snapshot matrix denoted as $\mathbf{U}^{(i)} = [\mathbf{u}_0^{(i)}, ..., \mathbf{u}_{N_t}^{(i)}] \in \mathbb{R}^{N_u \times (N_t+1)}$. Concatenating snapshot matrices corresponding to all training sample points gives a full snapshot matrix $\mathbf{U} \in \mathbb{R}^{N_u \times (N_t+1)N_\mu}$

$$\mathbf{U} = \left[\mathbf{U}^{(1)}, ..., \mathbf{U}^{(N_\mu)}\right]. \tag{8.4}$$

8.3.1 Autoencoders for nonlinear dimensionality reduction

An autoencoder [44, 45] is a special architecture of deep neural networks (DNNs) designed for dimensionality reduction or representation learning. As shown in Fig. 2.5, an autoencoder consists of an encoder function $\phi_e(\cdot;\theta_{\text{enc}}) : \mathbb{R}^{N_u} \to \mathbb{R}^{N_z}$ and a decoder function $\phi_d(\cdot;\theta_{\text{dec}}) : \mathbb{R}^{N_z} \to \mathbb{R}^{N_u}$, such that

$$\mathbf{z}_n^{(i)} = \phi_e(\mathbf{u}_n^{(i)};\theta_{\text{enc}}), \tag{8.5a}$$

$$\hat{\mathbf{u}}_n^{(i)} = \phi_d(\mathbf{z}_n^{(i)};\theta_{\text{dec}}) \tag{8.5b}$$

where $\mathbf{u}_n^{(i)} \in \mathbb{R}^{N_u}$ denotes the solution of a sampling point $\mu^{(i)} \in \mathscr{D}^h$, $i \in \mathbb{N}(N_\mu)$, at the n-th time step; $N_z \ll N_u$ is the latent dimension, θ_{enc} and θ_{dec} are trainable parameters of the encoder and the deconder, respectively; $\hat{\mathbf{u}}_n^{(i)} \in \mathbb{R}^{N_u}$ is the output of the autoencoder, a reconstruction of the original input $\mathbf{u}_n^{(i)}$. With the latent dimension N_z much smaller than the input dimension N_u, the encoder ϕ_e is trained to compress the high-dimensional input $\mathbf{u}_n^{(i)}$ and learn a low-dimensional representation, denoted as a latent variable $\mathbf{z}_n^{(i)} \in \mathbb{R}^{N_z}$, whereas the decoder ϕ_d reconstructs the input data by mapping the latent variable back to the high-dimensional space, as illustrated in Fig. 2.5.

Let us denote $\mathbf{Z}^{(i)} = [\mathbf{z}_0^{(i)},...,\mathbf{z}_{N_t}^{(i)}] \in \mathbb{R}^{N_z \times (N_t+1)}$ as the matrix of latent variables at all time steps of the sampling point $\mu^{(i)}$ and $\mathbf{Z} = \left[\mathbf{Z}^{(1)},...,\mathbf{Z}^{(N_\mu)}\right] \in \mathbb{R}^{N_z \times (N_t+1)N_\mu}$ as the full latent variable matrix of all sampling points in the parameter space. The corresponding reconstructed full snapshot matrix is denoted by $\hat{\mathbf{U}} = \left[\hat{\mathbf{U}}^{(1)},...,\hat{\mathbf{U}}^{(N_\mu)}\right] \in \mathbb{R}^{N_u \times (N_t+1)N_\mu}$, where $\hat{\mathbf{U}}^{(i)} = [\hat{\mathbf{u}}_0^{(i)},...,\hat{\mathbf{u}}_{N_t}^{(i)}] \in \mathbb{R}^{N_u \times (N_t+1)}$, $i \in \mathbb{N}(N_\mu)$, obtained from Eq. (8.5). The optimal trainable parameters of the autoencoder (θ_{enc} and θ_{dec} from Eq. (8.5)) are obtained by minimizing the loss function:

$$\mathscr{L}_{recon} := ||\mathbf{U} - \hat{\mathbf{U}}||_{L_2}^2. \tag{8.6}$$

Due to symmetric architectures of the encoder and the decoder in the autoencoder, the encoder architecture is used to denote the autoencoder architecture for simplicity. For example, the encoder architecture 6-4-3 denotes that there are 6, 4, and 3 artificial neurons in the input layer, the hidden layer, and the embedding layer that outputs the latent variables, respectively. As such, the decoder architecture in this case is 3-4-6 and the corresponding autoencoder architecture is 6-4-3-4-6.

8.3.2 Latent-space dynamics identification

Instead of learning complex physical dynamics of high-dimensional data, a dynamics identification (DI) model is introduced to capture dynamics of the autoencoder-discovered low-dimensional representation associated with the high-dimensional data. Therefore, the problem of the high-dimensional dynamical system in Eq. (8.1) is reduced to

$$\frac{d\mathbf{z}(t;\mu)}{dt} = \psi_{DI}(\mathbf{z},t;\mu), \quad t \in [0,T], \tag{8.7}$$

where $\mathbf{z}(t;\mu) = \phi_e(\mathbf{u}(t;\mu)) \in \mathbb{R}^{N_z}$ and ϕ_e denotes the encoder function introduced in Section 8.3.1. The dynamics of $\mathbf{u}$ can be reconstructed by using the decoder function: $\hat{\mathbf{u}}(t;\mu) = \phi_d(\mathbf{z}(t;\mu))$.

Given the discrete latent variable matrix $\mathbf{Z}^{(i)} = [\mathbf{z}_0^{(i)}, ..., \mathbf{z}_{N_t}^{(i)}] \in \mathbb{R}^{N_z \times (N_t+1)}$ of a sampling point $\mu^{(i)}$, $i \in \mathbb{N}(N_\mu)$, the governing dynamical function ψ_{DI} is approximated by a user-defined library of candidate basis functions $\Theta(\mathbf{Z}^{(i)T})$, expressed as

$$\dot{\mathbf{Z}}^{(i)T} \approx \dot{\hat{\mathbf{Z}}}^{(i)T} = \Theta(\mathbf{Z}^{(i)T})\Xi^{(i)}, \tag{8.8}$$

where $\Theta(\mathbf{Z}^{(i)T}) = [b_1(\mathbf{Z}^{(i)T}), b_2(\mathbf{Z}^{(i)T}), ..., b_{N_b}(\mathbf{Z}^{(i)T})] \in \mathbb{R}^{(N_t+1)\times N_l}$ has N_b candidate basis functions to capture the latent space dynamics, e.g., polynomial, trigonometric, and exponential functions; N_l denotes the number of columns of the library matrix, determined by the choice of the basis functions, see [9] for more details; $\Xi^{(i)} = [\xi_1^{(i)}, \xi_2^{(i)}, ..., \xi_{N_z}^{(i)}] \in \mathbb{R}^{N_l \times N_z}$ is an associated coefficient matrix.

In gLaSDI, the autoencoder and the DI model are trained simultaneously and interactively to identify simple and smooth latent-space dynamics. Therefore, $\dot{\mathbf{Z}}^{(i)T} = [\dot{\mathbf{z}}_0^{(i)T}, ..., \dot{\mathbf{z}}_{N_t}^{(i)T}]^T \in \mathbb{R}^{(N_t+1)\times N_z}$ in Eq. (8.8) is obtained by applying the chain rule and automatic differentiation (AD) [202] to the encoder network, i.e.,

$$\dot{\mathbf{z}}_n^{(i)} = \left(\nabla_{\mathbf{u}} \mathbf{z}_n^{(i)}\right) \dot{\mathbf{u}}_n^{(i)} = \nabla_{\mathbf{u}} \phi_e(\mathbf{u}_n^{(i)}) \dot{\mathbf{u}}_n^{(i)}, \quad n = 0, ..., N_t. \tag{8.9}$$

To ensure the consistency on the identified latent-dynamics, $\dot{\mathbf{Z}}$ and $\dot{\hat{\mathbf{Z}}}$, the following loss function is constructed, which imposes a constraint on the trainable parameters of the encoder (θ_{enc} via Eq. (8.9)) and the DI models ($\{\Xi^{(i)}\}_{i\in\mathbb{N}(N_\mu)}$ via Eq. (8.8)),

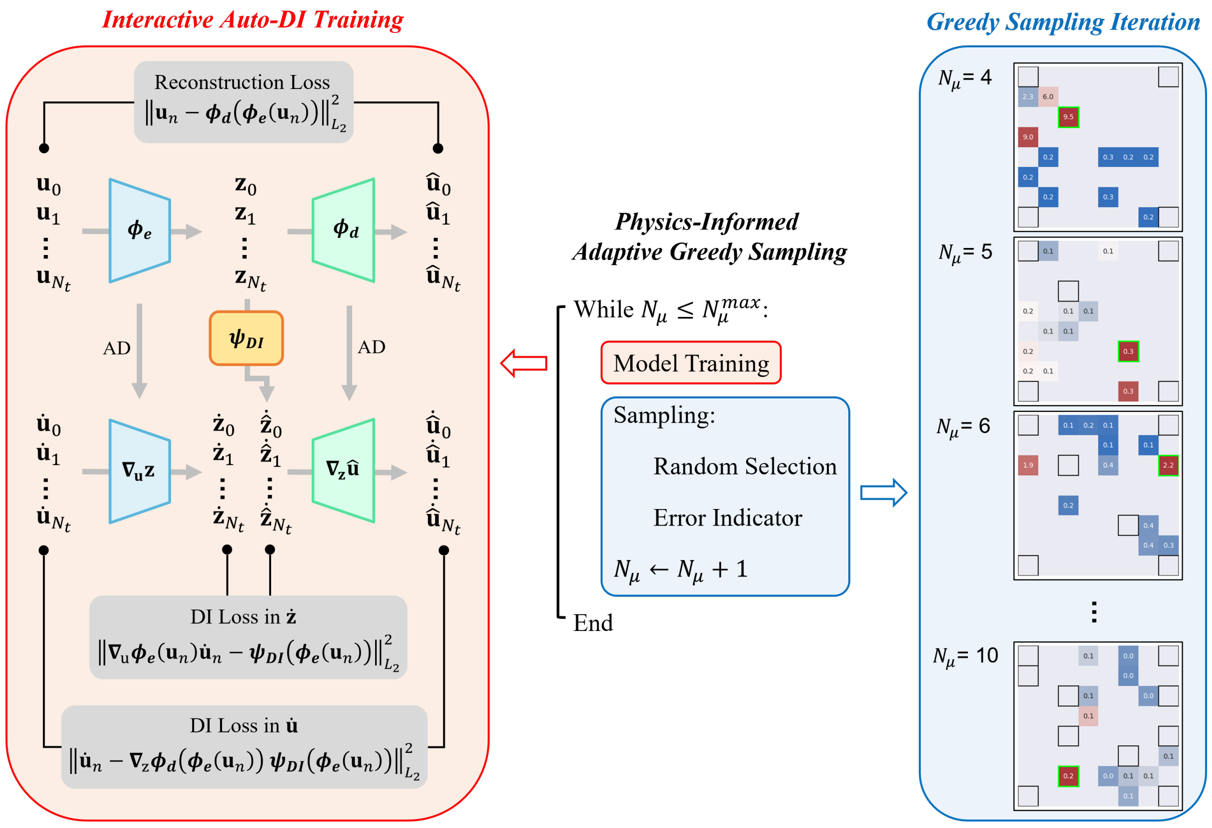
$$\mathscr{L}_{\dot{\mathbf{z}}} := ||\dot{\mathbf{Z}} - \dot{\hat{\mathbf{Z}}}||_{L_2}^2, \tag{8.10}$$

with

$$\dot{\mathbf{Z}} = \left[\dot{\mathbf{Z}}^{(1)}, ..., \dot{\mathbf{Z}}^{(N_\mu)}\right] \in \mathbb{R}^{N_z \times (N_t+1)N_\mu} \tag{8.11a}$$

$$\dot{\hat{\mathbf{Z}}} = \left[\dot{\hat{\mathbf{Z}}}^{(1)}, ..., \dot{\hat{\mathbf{Z}}}^{(N_\mu)}\right] \in \mathbb{R}^{N_z \times (N_t+1)N_\mu}. \tag{8.11b}$$

The loss function in $\dot{\mathbf{z}}$ also enables identification of simple latent-dynamics when simple DI model is prescribed, which will be demonstrated in Section 8.4.2-8.4.3. Note that the local DI models are considered to be point-wise (see the detailed description about point-wise and region-based DI models in [9]), which means each local DI model is associated with a distinct sampling point in the parameter space. Hence, each sampling point has an associated DI coefficient matrix.



Interactive Auto-DI Training
Reconstruction Loss $\|\mathbf{u}_n - \phi_d(\phi_e(\mathbf{u}_n))\|_{L_2}^2$
$\mathbf{u}_0$ $\mathbf{u}_1$ $\mathbf{u}_{N_t}$
ϕ_e
$\mathbf{z}_0$ $\mathbf{z}_1$ $\mathbf{z}_{N_t}$
ϕ_d
$\hat{\mathbf{u}}_0$ $\hat{\mathbf{u}}_1$ $\hat{\mathbf{u}}_{N_t}$
AD
ψ_{DI}
AD
$\dot{\mathbf{u}}_0$ $\dot{\mathbf{u}}_1$ $\dot{\mathbf{u}}_{N_t}$
$\nabla_\mathbf{u}\mathbf{z}$
$\dot{\mathbf{z}}_0$ $\dot{\mathbf{z}}_1$ $\dot{\mathbf{z}}_{N_t}$
$\dot{\hat{\mathbf{z}}}_0$ $\dot{\hat{\mathbf{z}}}_1$ $\dot{\hat{\mathbf{z}}}_{N_t}$
$\nabla_\mathbf{z}\hat{\mathbf{u}}$
$\dot{\hat{\mathbf{u}}}_0$ $\dot{\hat{\mathbf{u}}}_1$ $\dot{\hat{\mathbf{u}}}_{N_t}$
DI Loss in $\dot{\mathbf{z}}$ $\|\nabla_\mathbf{u}\phi_e(\mathbf{u}_n)\dot{\mathbf{u}}_n - \psi_{DI}(\phi_e(\mathbf{u}_n))\|_{L_2}^2$
DI Loss in $\dot{\mathbf{u}}$ $\|\dot{\mathbf{u}}_n - \nabla_\mathbf{z}\phi_d(\phi_e(\mathbf{u}_n))\,\psi_{DI}(\phi_e(\mathbf{u}_n))\|_{L_2}^2$
Physics-Informed Adaptive Greedy Sampling
While $N_\mu \leq N_\mu^{max}$:
Model Training
Sampling:
Random Selection
Error Indicator
$N_\mu \leftarrow N_\mu + 1$
End
Greedy Sampling Iteration
N_μ= 4
N_μ= 5
N_μ= 6
N_μ= 10

Figure 8.1. Schematic of the gLaSDI algorithm.

To further enhance the accuracy of the physical dynamics predicted by the decoder, a loss function is constructed to ensure the consistency of between the predicted dynamics gradients and the gradients of the solution data, in addition to the reconstruction loss function in Eq. (8.6). The enhancement is demonstrated in Section 8.4.1. The predicted dynamics gradients, $\dot{\hat{\mathbf{u}}}$, can be calculated from $\mathbf{u}$ by following the path: $\mathbf{u} \to \mathbf{z} \to \dot{\hat{\mathbf{z}}} \to \dot{\hat{\mathbf{u}}}$, as illustrated in Fig. 8.1, and applying the chain rule and AD to the decoder network,

$$\begin{aligned} \dot{\hat{\mathbf{u}}}_n^{(i)} &= \frac{\partial \hat{\mathbf{u}}_n^{(i)}}{\partial \mathbf{z}_n^{(i)}} \cdot \frac{\partial \mathbf{z}_n^{(i)}}{\partial t} && (8.12a) \\ &= \nabla_{\mathbf{z}} \phi_d \big(\phi_e (\mathbf{u}_n^{(i)}) \big) \cdot \psi_{DI}(\mathbf{z}_n^{(i)}) && (8.12b) \\ &= \nabla_{\mathbf{z}} \phi_d \big(\phi_e (\mathbf{u}_n^{(i)}) \big) \cdot \Theta(\phi_e(\mathbf{u}_n^{(i)})^T) \Xi^{(i)}. \quad n = 0, ..., N_t, && (8.12c) \end{aligned}$$

which involves all trainable parameters, including the encoder (θ_{enc}), the decoder (θ_{dec}), and the DI models ($\{\Xi^{(i)}\}_{i \in \mathbb{N}(N_\mu)}$). The loss function in $\dot{\mathbf{u}}$ is defined as

$$\mathscr{L}_{\dot{\mathbf{u}}} := ||\dot{\mathbf{U}} - \dot{\hat{\mathbf{U}}}||_{L_2}^2. \tag{8.13}$$

with

$$\begin{aligned} \dot{\mathbf{U}} &= \big[\dot{\mathbf{U}}^{(1)}, ..., \dot{\mathbf{U}}^{(N_\mu)} \big] \in \mathbb{R}^{N_u \times (N_t+1) N_\mu} && (8.14a) \\ \dot{\hat{\mathbf{U}}} &= \big[\dot{\hat{\mathbf{U}}}^{(1)}, ..., \dot{\hat{\mathbf{U}}}^{(N_\mu)} \big] \in \mathbb{R}^{N_u \times (N_t+1) N_\mu}. && (8.14b) \end{aligned}$$

Therefore, the loss function of the *interactive auto-DI training* consists of three different loss terms, i.e., the reconstruction loss of the autoencoder in Eq. (8.6), the DI loss in $\dot{\mathbf{z}}$ in Eq. (8.10), and the DI loss in $\dot{\mathbf{u}}$ in Eq. (8.13). They are combined as a linear combination:

$$\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}} + \beta_2 \mathscr{L}_{\dot{\mathbf{u}}}, \tag{8.15}$$

where β_1 and β_2 denote the regularization parameters to balance the scale and contributions from the loss terms. A schematics of the *interactive auto-DI training* is shown in Fig. 8.1.

Compared with a global DI model that captures global latent-space dynamics of all sampling points in the parameter space $\mathscr{D}^h$, each local DI model in the proposed framework is associated with one sampling point and thus captures local latent-space dynamics more accurately. Further, an efficient k-nearest neighbor (KNN) convex interpolation scheme, which will be introduced in the next subsection, is employed to exploit the local latent-space dynamics captured by the local DI models for an improved prediction accuracy, which will be demonstrated in Section 8.4.1.

8.3.3 k-nearest neighbors convex interpolation

To exploit the local latent-space dynamics captured by the local DI models for enhanced parameterization and efficiency, a KNN convexity-preserving partition-of-unity interpolation scheme is employed. The interpolation scheme utilizes Shepard function [206] or inverse distance weighting, which has been widely used in data fitting and function approximation with positivity constraint [6, 207–209]. Compared with other interpolation techniques, such as the locally linear embedding [38, 196] and the radial basis function interpolation [9], which require optimization to obtain interpolation weights and thus more computational cost, the employed KNN Shepard interpolation is more efficient while preserving convexity.

Given a testing parameter $\mu \in \mathscr{D}$, the DI coefficient matrix Ξ is obtained by a convex interpolation of coefficient matrices of its k-nearest neighbors (existing sampling points), expressed as

$$\Xi_{interp} = \mathscr{I}\left(\{\Psi^{(i)}(\mu); \Xi^{(i)}\}_{i \in \mathscr{N}_k(\mu)}\right) = \sum_{i \in \mathscr{N}_k(\mu)} \Psi^{(i)}(\mu) \Xi^{(i)}, \tag{8.16}$$

where Ξ_{interp} is the interpolated DI coefficient matrix of the testing parameter μ, $\mathscr{N}_k(\mu)$ is the index set of the k-nearest neighbor points of μ selected from $\mathbb{D} \subseteq \mathscr{D}^h$ that contains the parameters of the training samples, and $\Xi^{(i)}$ is the coefficient matrix of the sampling point $\mu^{(i)}$. The selection of the k-nearest neighbors is based on the Euclidean distance between the testing parameter and the training parameters, $||\mu - \mu^{(i)}||_{L_2}$. The interpolation functions are defined as

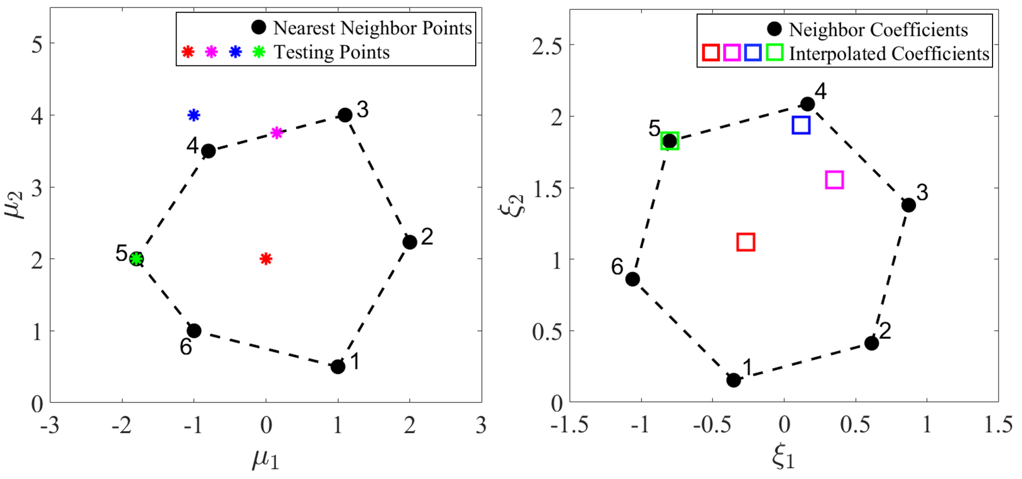
$$\Psi^{(i)}(\mu) = \frac{\phi(\mu - \mu^{(i)})}{\sum_{j=1}^{k} \phi(\mu - \mu^{(j)})}, \tag{8.17}$$

where k is the number of nearest neighbors. In Eqs. (8.16) and (8.17), ϕ is a positive kernel function representing the weight on the data set $\{\Xi^{(i)}\}_{i \in \mathscr{N}_k(\mu)}$, and $\mathscr{I}$ denotes the interpolation operator that constructs shape functions with respect to μ and its neighbors. It should be noted that these functions satisfy a partition of unity, i.e., $\sum_{i \in \mathscr{N}_k(\mu)} \Psi^{(i)}(\mu) = 1$ for transformation objectivity. Furthermore, they are convexity-preserving when the kernel function ϕ is a positive function. Here, an inverse distance function is used as the kernel function

$$\phi(\mu - \mu^{(i)}) = \frac{1}{||\mu - \mu^{(i)}||_{L_2}^2}. \tag{8.18}$$

If the testing parameter point overlaps with one of the nearest neighbor points, i.e., $\mu = \mu^{(j)}$, $j \in \mathscr{N}_k(\mu)$, then $\Psi^{(j)}(\mu) = 1$ and $\Psi^{(i)}(\mu) = 0$, $\forall i \in \mathscr{N}_k(\mu)$ and $i \neq j$, resulting in $\Xi_{interp} = \Xi^{(j)}$, which is expected.

For a better visualization and a clear demonstration in a two-dimensional domain, we consider convex interpolation of two components from the coefficient matrix, i.e., ξ_1 and ξ_2, as shown in Fig. 8.2(b). Fig. 8.2(a) shows four testing parameter points denoted by asterisks and their 6 nearest neighbor parameter points denoted by solid black dots. The testing parameter points are at different locations relative to the convex hull (depicted by the black dash line) formed by the nearest neighbor points. The interpolation functions are obtained using Eqs. (8.17-8.18) based on the Euclidean distance between the testing points and the nearest neighbor points, which are then used to interpolate the coefficients of the testing points from the coefficients of the nearest neighbors by using Eq. (8.16), as shown in Fig. 8.2(b). It can be seen that the interpolated coefficients are all located within the convex hull (depicted by the black dash line) formed by the nearest neighbors' coefficients, showing the desired convexity-preserving capability, which allows existing local DI models to be leveraged for prediction of latent-space dynamics of testing parameters. When the testing parameter point (the green asterisk) overlaps with the nearest neighbor point 5, the coefficient of the testing point is identical with that of the nearest neighbor point 5. The convex interpolation is simple and efficient as the interpolation functions in Eq. (8.17) can be constructed easily in the parameter space.



(a) **(b)**

Figure 8.2. Demonstration of the convexity-preserving interpolation in Eq. (8.16): (a) The four asterisks denote the testing parameter points while the black solid dots denote the nearest neighbor parameter points of the testing parameter points. The black dash line depicts a locally convex hull formed by the nearest neighbor points. The interpolation functions are obtained using Eqs. (8.17-8.18) based on the Euclidean distance between the testing points and nearest neighbor points. (b) The black solid dots denote the coefficients associated with the nearest neighbor parameter points in (a), which are used to interpolate the coefficients (squares) associated with the testing points in (a) by using the convex interpolation scheme in Eq. (8.16). The black dash line depicts a locally convex hull formed by the nearest neighbors' coefficients.

8.3.4 Physics-informed adaptive greedy sampling

In the proposed algorithm, the training sample points are determined on the fly by a physics-informed adaptive greedy sampling algorithm to maximize parameter space exploration and achieve optimal model performance. To this end, a sampling procedure is integrated with an error indicator is needed. The most accurate error indicator would be actual relative error that is computed with high-fidelity solutions. However, requiring high-fidelity solution is computationally expensive, leading to undesirable training cost and time. Therefore, we adopt a physics-informed residual-based error indicator, which does not require high-fidelity solutions. The residual-based error indicator is defined in Eq. (8.21). The residual-based error indicator has a positive correlation with the maximum relative error and can be efficiently computed based only on predicted gLaSDI solutions. For example, see Figure 8.3. The physics-informed residual-based error indicator is integrated into an adaptive greedy sampling algorithm with a multi-level random-subset evaluation strategy. Various termination criteria for the adaptive greedy sampling are discussed in the following subsection.

Adaptive greedy sampling procedure

To address the issues of parameter dependency of local latent-space dynamics efficiently and effectively, an adaptive greedy sampling procedure is applied to construct a database $\mathscr{D}\mathscr{B} = \{\mathbf{U}^{(i)}\}_{i=1}^{N_\mu}$ on the fly during offline training, which corresponds to a set of sampled parameters $\mathbb{D} = \{\mu^{(i)}\}_{i=1}^{N_\mu}$ from the discrete parameter space $\mathscr{D}^h$, $N_\mu < N_{\mathscr{D}} = |\mathscr{D}^h|$; $\mathbf{U}^{(i)} = [\mathbf{u}_0^{(i)}, ..., \mathbf{u}_{N_t}^{(i)}] \in \mathbb{R}^{N_u \times (N_t+1)}$ is the high-fidelity solution of the parameter $\mu^{(i)}$.

The database is first initialized with a small set of parameters located, e.g., at the corners of the boundaries or at the center of the parameter space. To enhance sampling reliability and quality, the model training is performed before greedy sampling, as illustrated in Fig. 8.1. Therefore, the adaptive greedy sampling is performed after every N_{up} epochs of training, which means a new training sample is added to the training database for every N_{up} epochs. At the v-th sampling iteration, a set of candidate parameters are considered and the parameter that maximizes an error indicator, $e\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big)$, is selected. The definition of the error indicator is introduced in the next section. The greedy sampling procedure is summarized as

$$\mu^* = \underset{\mu \in \mathbb{D}'}{\text{argmax}}\ e\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big), \tag{8.19a}$$

$$\mathscr{DB}_v = \{\mathscr{DB}_{v-1}, \mathbf{U}^*)\}, \tag{8.19b}$$

$$\mathbb{D}_v = \{\mathbb{D}_{v-1}, \mu^*\}, \tag{8.19c}$$

where $\mathbb{D}^{'} \subseteq \mathscr{D}^h$ denotes a set of $N^{'} \leq N_{\mathscr{D}}$ candidate parameters and $\mathbb{D}^{'} \cap \mathbb{D}_{v-1} = \emptyset$; $\mathbb{D}_{v-1}$ contains the parameters associated with $\mathscr{DB}_{v-1}$; μ^* denotes the selected parameter and $\mathbf{U}^*$ is the corresponding high-fidelity solution. The iterations of greedy sampling continue until a certain criterion is reached, which will be discussed in the following subsection.

Physics-informed residual-based error indicator

Given an approximate gLaSDI solution, $\hat{\mathbf{U}}(\mu) = [\hat{\mathbf{u}}_0(\mu), ..., \hat{\mathbf{u}}_{N_t}(\mu)]$, of the corresponding high-fidelity true solution, $\mathbf{U}(\mu) = [\mathbf{u}_0(\mu), ..., \mathbf{u}_{N_t}(\mu)]$, the maximum relative error, $e^{max}\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big)$, is defined as

$$e^{max}\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big) = \max_{n \in \mathbb{N}_t} \left(\frac{||\mathbf{u}_n(\mu) - \hat{\mathbf{u}}_n(\mu)||_{L_2}}{||\mathbf{u}_n(\mu)||_{L_2}} \right). \tag{8.20}$$

The maximum relative error as an error indicator provides the most accurate guidance to the greedy sampling procedure, However, the evaluation of $e^{max}\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big)$ is computationally inefficient because of the requirement of the high-fidelity true solution. To ensure effective and efficient greedy sampling, the error indicator needs to satisfy the following criteria: (i) It must be positively correlated with the maximum relative error measure, as demonstrated in Fig. 8.3; (ii) The evaluation is computationally efficient, i.e., it must not involve any high-fidelity solution. A computationally feasible error indicator based on the residual of the governing equation is employed in this study, defined as

$$e^{res}\big(\hat{\mathbf{U}}(\mu)\big) = \frac{1}{N_{ts}+1}\sum_{n=0}^{N_{ts}} ||\mathbf{r}(\hat{\mathbf{u}}_n;\hat{\mathbf{u}}_{n-1},\mu)||_{L_2} \tag{8.21}$$

where the residual function $\mathbf{r}(\hat{\mathbf{u}}_n;\hat{\mathbf{u}}_{n-1},\mu)$ is defined in Eq. (8.3) and $N_{ts} < N_t$. The residual error indicator satisfies the aforementioned two conditions. For example, note that the evaluation of the error indicator requires only the predicted gLaSDI solutions and the fact that we use $N_{ts} < N_t$ further enhances computational efficiency of the error indicator. In this paper, $N_{ts}/N_t \approx 0.1$ is used. Furthermore, the adopted error indicator is positively correlated with the maximum relative error, which is demonstrated in Figure 8.3. Note also that it is physics-informed as it is based on the residual of the discretized governing equations, which embeds physics (Eq. (8.3).

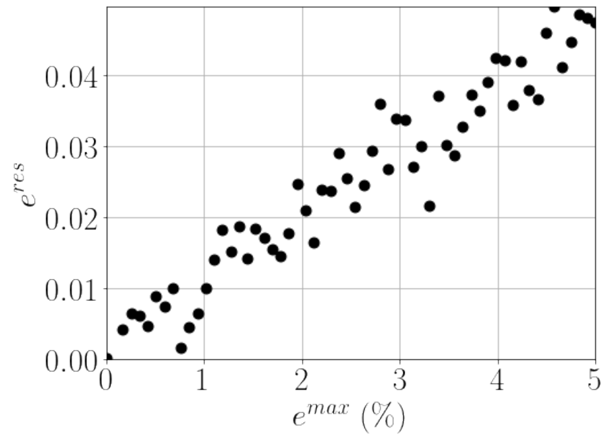


Figure 8.3. A demonstration to show positive correlation between the residual-based error indicator e^{res} and the maximum relative error e^{max}. This indicates the computationally efficient e^{res} can replace the computationally expensive e^{max}. Each black-filled circle represents one evaluated sample.

Finally, the next parameter to be sampled is determined by

$$\mu^* = \underset{\mu\in\mathbb{D}'}{\operatorname{argmax}}\; e^{res}\big(\hat{\mathbf{U}}(\mu)\big). \tag{8.22}$$

Termination criteria

The adaptive greedy sampling procedure is terminated until one of the following criteria is reached: (i) a prescribed maximum number of sampling points (local DI models), (ii) the maximum allowable training iterations, or (iii) a prescribed target tolerance of the maximum relative error.

As the training cost increases with the number of sampling points and the number of training iterations, criteria (i) and (ii) are considered if training efficiency is preferable. On the other hand, if one expects the optimal model performance, criterion (iii) is more suitable as it offers a guidance of the model accuracy in the parameter space.

Since the maximum relative error is estimated by the residual-based error indicator, as described in Section 8.3.4, criterion (iii) only provides an estimated model performance. To alleviate this issue, we exploit the ratio between the maximum relative error and the residual-based error indicator to approximate the correct target relative error. For example, at v-th sampling iteration, the model is evaluated to obtain the maximum relative errors and the residual-based errors of all sampled parameters:

$$\mathbf{E}_v^{max} = \{e^{max}\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big)\}_{\mu \in \mathbb{D}_v}, \tag{8.23a}$$

$$\mathbf{E}_v^{res} = \{e^{res}\big(\hat{\mathbf{U}}(\mu)\big)\}_{\mu \in \mathbb{D}_v}, \tag{8.23b}$$

where $e^{max}\big(\mathbf{U}(\mu), \hat{\mathbf{U}}(\mu)\big)$ and $e^{res}\big(\hat{\mathbf{U}}(\mu)\big)$ are calculated from Eq. (8.20) and Eq. (8.21), respectively. Note that $\mathbf{E}_v^{max}$ can be obtained because the database $\mathscr{D}\mathscr{B}_v$ contains high-fidelity solutions of all sampled parameters in $\mathbb{D}_v$. Then, linear correlation coefficients (k^*, b^*) between $\mathbf{E}_v^{res}$ and $\mathbf{E}_v^{max}$ are obtained by

$$(k^*, b^*) = \underset{k,b}{\arg\min} ||\mathbf{E}_v^{max} - \big(k \cdot \mathbf{E}_v^{res} + b\big)||_{L_2}^2. \tag{8.24}$$

Finally, the estimated maximum relative error is obtained by

$$e_v^{max} = k^* \cdot \max(\mathbf{E}_v^{res}) + b^*. \tag{8.25}$$

As the correlation between $\mathbf{E}_v^{res}$ and $\mathbf{E}_v^{max}$ of all sampled parameters in $\mathbb{D}_v$ is used to estimate the maximum relative error e_v^{max}, it improves the termination guidance of the greedy sampling procedure in order to achieve the target tol, leading to more reliable reduced-order models. This provides some level of confidence in the accuracy of the trained gLaSDI.

Multi-level random-subset evaluation

To further accelerate the greedy sampling procedure, a two-level random-subset evaluation strategy is adopted in this study. At the v-th sampling iteration of the first level, a subset, $\mathbb{D}^{'}$, of parameters is randomly selected from the parameter space, i.e., $\mathbb{D}^{'} \subseteq \mathscr{D}^h$ and $\mathbb{D}^{'} \cap \mathbb{D}_{v-1} = \emptyset$. A small subset size $N_{subset} = |\mathbb{D}^{'}|$ is considered in the first-level random subset selection so that the error evaluation of the parameters in the subset is efficient. When the tolerance of the error indicator tol is reached the greedy sampling procedure moves to the second-level random subset evaluation, where the subset size N_{subset} doubles. The greedy sampling procedure continues until the prescribed termination criterion is reached, see Algorithm 2 for more details.

8.3.5 gLaSDI off-line stage

The ingredients mentioned in earlier sections are integrated into the proposed greedy latent-space dynamics identification model. The training procedure of the gLaSDI model is summarized in Algorithm 1.

8.3.6 gLaSDI on-line stage

After the gLaSDI model is trained by Algorithm 1, the physics-informed greedy sample parameter set $\mathbb{D}$, the autoencoder parameters, and a set of local DI model parameters are obtained. The trained gLaSDI model can then be applied to efficiently predict dynamical solutions given

Algorithm 1. Training of the gLaSDI model

Input: An initial parameter set $\mathbb{D}_0 \subseteq \mathscr{D}^h$ and the associated database $\mathscr{DB}_0$; an initial random subset size N_{subset}; a greedy sampling frequency N_{up}; the number of nearest neighbors k for KNN convex interpolation; and one of the three terminal criteria:

- a target tolerance of the maximum relative error tol
- the maximum number of sampling points N_{μ}^{max}
- a maximum number of training epochs N_{epoch}

Note that N_{epoch} is often used together with the error tolerance to avoid excessive training iterations in the case where the prescribed tolerance may not be achieved.

Output: gLaSDI sampled parameter set $\mathbb{D}_v$ and the associated database $\mathscr{DB}_v$; autoencoder parameters; θ_{enc} and θ_{dec}; DI model coefficients $\{\Xi^{(i)}\}_{i \in \mathscr{N}_{\mathbb{D}_v}}$, where $\mathscr{N}_{\mathbb{D}_v}$ contains indices of parameters in $\mathbb{D}_v$

```
1: Set $v = 1$                                              ▷ iteration counter
2: Set $w = 1$                        ▷ level counter for random-subset selection
3: Set epoch = 1                                       ▷ training epoch counter
4: while epoch $\leq N_{epoch}$ do                      ▷ gLaSDI training iterations
5:     Update $\theta_{enc}$, $\theta_{dec}$, $\{\Xi^{(i)}\}_{i \in \mathscr{N}_{\mathbb{D}_{v-1}}}$ by minimizing the gLaSDI loss function in Eq. (8.15)
   with $\mathscr{DB}_{v-1}$ and $\mathbb{D}_{v-1}$
6:     if epoch $\bmod N_{up} = 0$ then                         ▷ greedy sampling
7:         Obtain $\mathbb{D}_v, \mathscr{DB}_v, e_v^{max}, N_{subset}, w$             ▷ algorithm 2
8:         $v \leftarrow v + 1$
9:     end if
10:    if $\left(e_v^{max} \leq tol \text{ and } w = 2\right)$ or $\left(|\mathbb{D}_v| > N_{\mu}^{max}\right)$ then
11:        break                                    ▷ terminate gLaSDI training
12:    end if
13:    epoch ← epoch + 1
14: end while
15: return $\mathbb{D}_v$, $\mathscr{DB}_v$, $\theta_{enc}$, $\theta_{dec}$, $\{\Xi^{(i)}\}_{i \in \mathscr{N}_{\mathbb{D}_v}}$
```

Algorithm 2. Greedy sampling with random-subset evaluation

Input: A parameter set $\mathbb{D}_{v-1} \subseteq \mathscr{D}^h$ and the associated database $\mathscr{DB}_{v-1}$; a target tolerance of the maximum relative error tol; a random-subset size N_{subset}; a random-subset level w; the number of nearest neighbors k for KNN convex interpolation

Output: updated parameter set $\mathbb{D}_v$; database $\mathscr{DB}_v$; estimated maximum relative error e_v^{max}; random-subset size N_{subset}; random-subset level w

```
 1: Select a random subset of parameters $\mathbb{D}^{'} \in \mathscr{D}^h$ with a size of $N_{subset}$
 2: for $\mu \in \mathbb{D}^{'}$ do                                        ▷ gLaSDI predictions
 3:     Obtain $\hat{\mathbf{U}}(\mu)$ from model evaluation                          ▷ algorithm 3
 4: end for
 5: Compute $e^{res}(\hat{\mathbf{U}}(\mu)),\ \forall \mu \in \mathbb{D}^{'}$ by Eq. (8.21)
 6: Obtain $\mu^*$ by solving Eq. (8.22)
 7: Obtain $\mathbf{U}(\mu^*)$ by solving Eq. (8.1) numerically
 8: $\mathscr{DB}_v \leftarrow \{\mathscr{DB}_{v-1}, \mathbf{U}(\mu^*)\}$                           ▷ update database
 9: $\mathbb{D}_v \leftarrow \{\mathbb{D}_{v-1}, \mu^*\}$                                 ▷ update the parameter set
10: Obtain $\{\hat{\mathbf{U}}(\mu)\}_{\mu \in \mathbb{D}_v}$                                     ▷ algorithm 3
11: Compute $\mathbf{E}_v^{max}$ by Eq. (8.23a)
12: Compute $\mathbf{E}_v^{res}$ by Eq. (8.23b)
13: Compute $e_v^{max}$ by Eqs. (8.24)-(8.25)
14: if $e_v^{max} \leq tol$ and $w < 2$ then
15:     $N_{subset} \leftarrow 2 \times N_{subset}$                          ▷ update the random subset size
16:     $w \leftarrow w + 1$
17: end if
18: return $\mathbb{D}_v, \mathscr{DB}_v, e_v^{max}, N_{subset}, w$
```

a testing parameter by using Algorithm 3. The prediction accuracy can be evaluated by the maximum relative error with respect to the corresponding true solutions using Eq. (8.20).

Algorithm 3. Evaluation of the gLaSDI model

```
Input: A testing parameter $\mu \in \mathscr{D}^h$; the gLaSDI sampled parameter set $\mathbb{D}$; the model coefficients $\theta_{enc}$; $\theta_{dec}$; $\{\Xi^{(i)}\}_{i \in \mathscr{N}_{\mathbb{D}}}$; and the number of nearest neighbors $k$ for KNN convex interpolation
Output: gLaSDI prediction $\hat{\mathbf{U}}(\mu)$
1: Search for KNN parameters based on the $L_2$ distance, $\mathscr{N}_k(\mu)$
2: Compute KNN convex interpolation functions by Eqs. (8.17-8.18)
3: Obtain $\Xi_{interp}$ for $\mu$ by convex interpolation in Eq. (8.16)
4: Compute initial latent variables $\mathbf{z}_0 = \phi_e(\mathbf{u}_0; \theta_{\text{enc}})$ in Eq. (8.5a)
5: Compute $\{\hat{\mathbf{z}}_n\}_{n=0}^{N_t}$ by Eq. (8.8)
6: Compute $\{\hat{\mathbf{u}}_n\}_{n=0}^{N_t}$ by Eq. (8.5b)
7: $\hat{\mathbf{U}}(\mu) \leftarrow [\hat{\mathbf{u}}_0, ..., \hat{\mathbf{u}}_{N_t}]$
8: return $\hat{\mathbf{U}}(\mu)$
```

8.4 Numerical results

The performance of gLaSDI is demonstrated by solving four numerical problems: one-dimensional (1D) Burgers' equation, two-dimensional (2D) Burgers' equation, nonlinear time-dependent heat conduction, and radial advection. The effects of the number of nearest neighbors k on the model performance are discussed. In each of the numerical examples, the gLaSDI's performance is compared with that of LaSDI, that is the one without adaptive greedy sampling. Note that the physics-informed adaptive greedy sampling can be performed on either a continuous parameter space ($\mathscr{D}$) or a discrete parameter space ($\mathscr{D}^h \subseteq \mathscr{D}$). In the following examples, a discrete parameter space ($\mathscr{D}^h$) is considered. For presented numerical experiments, we always start with the initial database $\mathscr{DB}_0$ and the associated parameter set $\mathbb{D}_0$ of four corner points of the parameter space.

The training and testing are performed on a NVIDIA V100 (Volta) GPU of the Livermore Computing Lassen system at the Lawrence Livermore National Laboratory, with 3,168 NVIDIA CUDA Cores and 64 GB GDDR5 GPU Memory. The open-source TensorFlow library [219] and the Adam optimizer [215] are employed for model training.

8.4.1 1D Burgers equation

A 1D parameterized inviscid Burgers equation is considered

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad x \in \Omega = [-3,3], \quad t \in [0,1], \tag{8.26a}$$

$$u(3,t;\mu) = u(-3,t;\mu). \tag{8.26b}$$

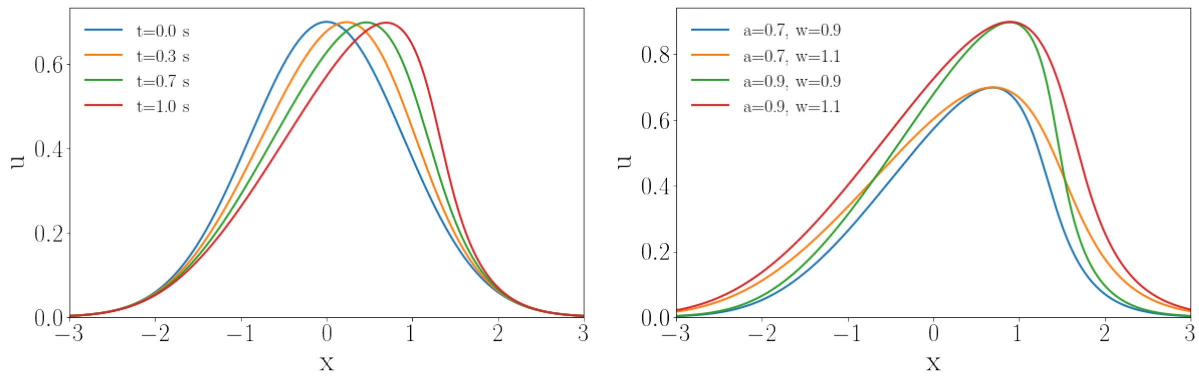
Eq. (8.26b) is a periodic boundary condition. The initial condition is parameterized by the amplitude a and the width w, defined as

$$u(x,0;\mu) = ae^{-\frac{x^2}{2w^2}}, \tag{8.27}$$

where $\mu = \{a,w\}$. A uniform spatial discretization with 1,001 discrete points and nodal spacing as dx =6/1,000 is applied. The first order spatial derivative is approximated by the backward difference scheme. A semi-discertized system characterized by the ODE described in Eq. (8.1) is obtained, which is solved by using the implicit backward Euler time integrator with a uniform time step of Δt =1/1,000 to obtain the full-order model solutions. The physical dynamics of the parameter case $(a = 0.7, w = 0.9)$ and the solution fields at the last time step of 4 different parameter cases, i.e., $(a = 0.7, w = 0.9)$, $(a = 0.7, w = 1.1)$, $(a = 0.9, w = 0.9)$, and $(a = 0.9, w = 1.1)$ are shown in Fig. 8.4.

Case 1: Effects of the number of nearest neighbors k

In the first example, the effects of the number of nearest neighbors k on model performance are investigated. The parameter space, $\mathscr{D}^h$, considered in this example is constituted by the parameters of the initial condition, including the width, $w \in [0.9, 1.1]$, and the amplitude, $a \in [0.7, 0.9]$, each with 21 evenly distributed discrete points in the respective parameter range, resulting in 441 parameter cases in total. The gLaSDI model is composed of an autoencoder with an architecture of 1,001-100-5 and linear DI models. The greedy sampling is performed



(a) $a = 0.7, w = 0.9$ **(b)** snapshots at the last time step

Figure 8.4. (a) Physical dynamics of the parameter case $(a = 0.7, w = 0.9)$; (b) The solution fields at the last time step of 4 different parameter cases: $(a = 0.7, w = 0.9)$, $(a = 0.7, w = 1.1)$, $(a = 0.9, w = 0.9)$, and $(a = 0.9, w = 1.1)$.

until the estimated maximum relative error of sampled parameter points is smaller than the target tolerance, $tol = 5\%$. An initial random subset size $N_{subset} = 64$ is used, around 20% of the size of $\mathscr{D}^h$. A two-level random-subset evaluation scheme is adopted, as described in Section 8.3.4. The maximum number of training epoch N_{epoch} is set to be 50,000.

Adaptive greedy sampling with $k = 1$

The greedy sampling frequency N_{up} is set to be 2,000. The training is performed by using Algorithm 1, where $k = 1$ is used for greedy sampling procedure (Algorithm 2), which means the gLaSDI model utilizes the DI coefficient matrix of the existing parameter that is closest to the randomly selected parameter to perform dynamics predictions. Fig. 8.5 shows the history of the loss function in a red solid line and the maximum residual-based error of the sampled parameter points in a blue solid line, demonstrating the convergence of the training. In Fig. 8.5, the first blue point indicates the initial state of the model where four corner points of the parameter space are sampled, while the last blue point indicates the final state of the model that satisfies the prescribed termination criterion and no sampling is performed. The blue points in-between indicate the epoch where greedy sampling is performed. At the end of the training, 22 parameter points are sampled and stored, including the initial 4 parameter points located at the four corners of the parameter space, which means 22 local DI models are constructed and trained in the gLaSDI model.

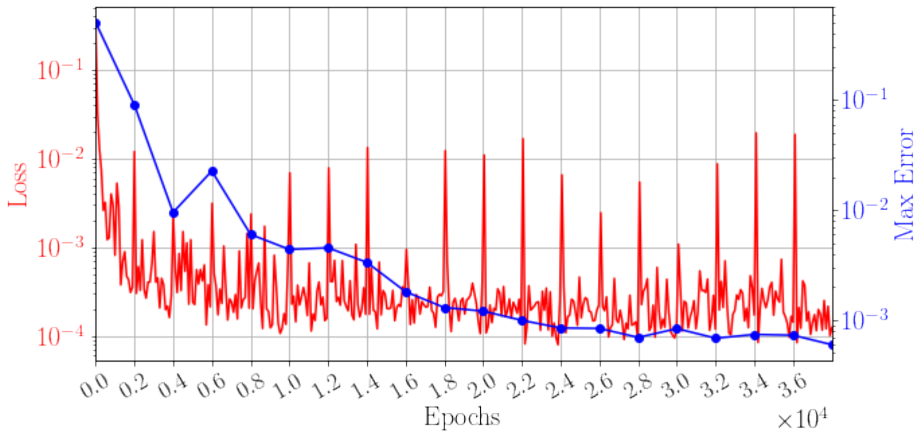


Figure 8.5. The history of the loss function and the maximum residual-based error of the sampled parameter points for the 1D Burgers problem. The KNN parameter, $k = 1$, is used for greedy sampling procedure during training.

After training, the gLaSDI model is applied for predictions by Algorithm 3, where different values of k are used for KNN convex interpolation of the DI coefficient matrix of the testing parameter. Fig. 8.6 shows the maximum relative errors in the parameter space $\mathscr{D}^h$ evaluated with different values of k. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled parameter points. The distance between the sampled parameter points located in the interior domain of $\mathscr{D}^h$ is relatively larger than that near the domain corners/boundaries. That means the interior sampled parameter points tend to have a larger *trust region* within which the model prediction accuracy is high. It can also be observed that model evaluation with $k > 1$ produces higher accuracy with the maximum relative error in $\mathscr{D}^h$ as around 2%, smaller than the target tolerance $tol = 5\%$, which is contributed by the KNN convex interpolation that exploits *trust region* of local DI models. Compared with the high-fidelity simulation based on an in-house

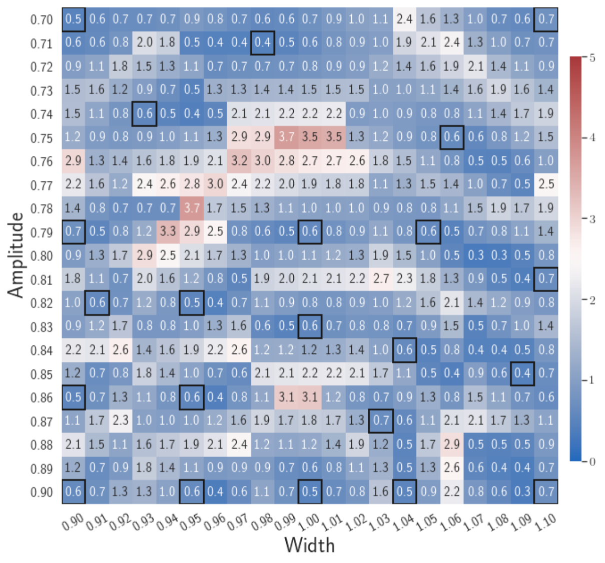
Python code, the gLaSDI model achieves 450$\times$ speed-up.

Adaptive greedy sampling with $k = 4$

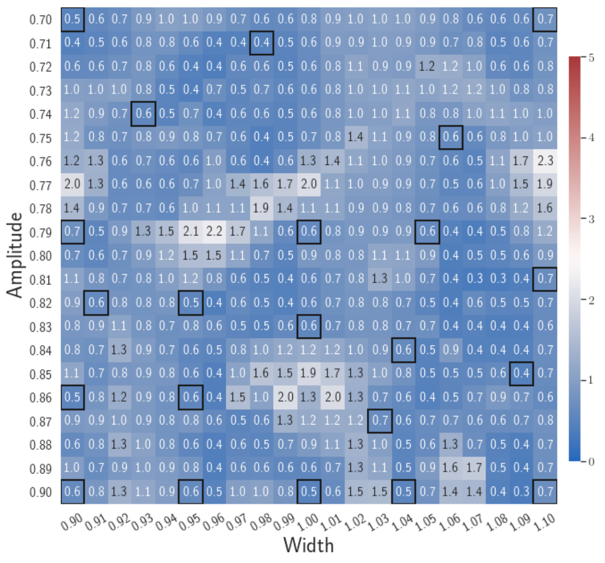
Fig. 8.7 shows the history of the loss function and the maximum residual-based error of the gLaSDI model trained with $k = 4$ for greedy sampling procedure. At the end of the training, 16 parameter points are sampled, including the initial four parameter points located at the 4 corners of the parameter space, which means 16 local DI models are constructed and trained in the gLaSDI model. Compared to the gLaSDI model trained with $k = 1$ as in Fig. 8.5, the training with $k = 4$ terminates faster with a sparser sampling to achieve the target tolerance of the maximum relative error, $tol = 5\%$, implying more efficient training. It is because a small k for greedy sampling procedure leads to a more conservative gLaSDI model with more local DIs, while a large k results in a more aggressive gLaSDI model with fewer local DIs as the trust region of local DI models are exploited by the KNN convex interpolation during training.

Fig. 8.8 shows the maximum relative errors in the parameter space $\mathscr{D}^h$ evaluated with different values of k. Similar to the results shown in Fig. 8.6, a larger k results in higher model accuracy. It is noted that a few violations of the target tolerance $tol = 5\%$ exist even when $k > 1$ is used for model evaluation. It shows that greedy sampling with $k > 1$ results in a more aggressive gLaSDI model with fewer local DIs and higher training efficiency at the cost of model accuracy. We also observed that the trained gLaSDI model achieved the best testing accuracy with $k = 4$, which implies there exists a certain $k > 1$ for optimal model performance.

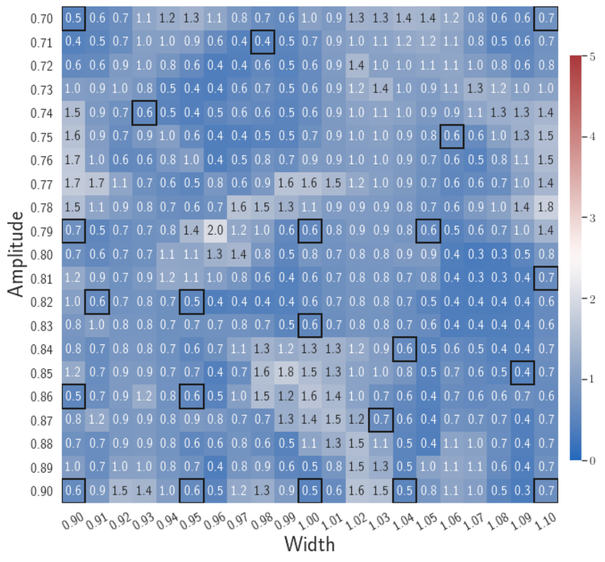
The comparison of these two tests shows that a small k for greedy sampling during training results in a more accurate gLaSDI model at the cost of training efficiency, and that using a $k > 1$ for model evaluation (testing) improves generalization performance of gLaSDI. In the following numerical examples, $k = 1$ is used for greedy sampling procedure during training of gLaSDI. The trained gLaSDI models are evaluated by different $k > 1$ and the the optimal testing



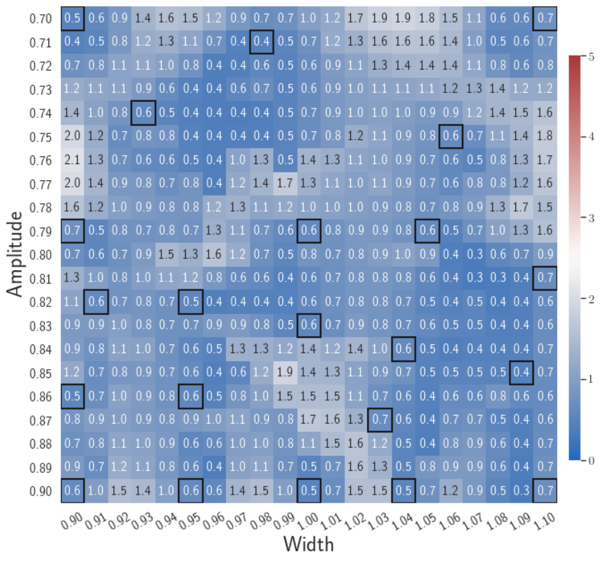
(a) $k = 1$ evaluation



(b) $k = 3$ evaluation



(c) $k = 4$ evaluation



(d) $k = 5$ evaluation

Figure 8.6. Maximum relative errors in the parameter space $\mathscr{D}^h$ evaluated with different values of k for KNN convex interpolation of the DI coefficient matrices of the testing parameters for 1D Burgers problem: (a) $k = 1$, (b) $k = 3$, (c) $k = 4$, (d) $k = 5$. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the parameter points sampled from training. $k = 1$ is used for greedy sampling procedure during training.

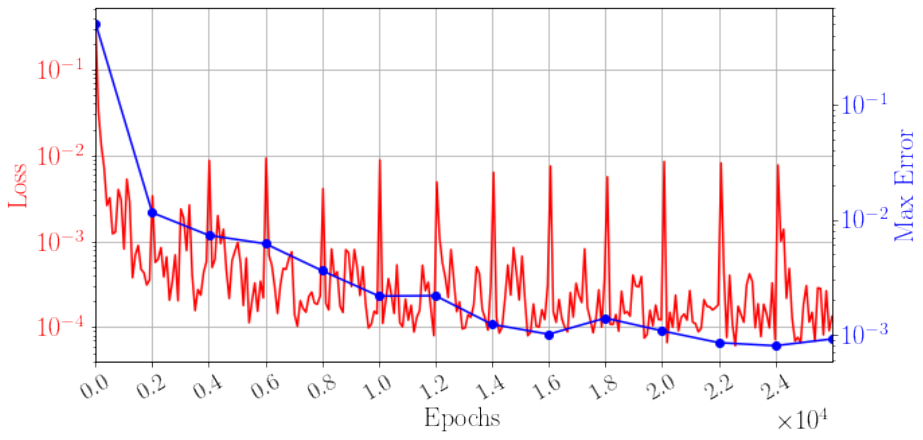


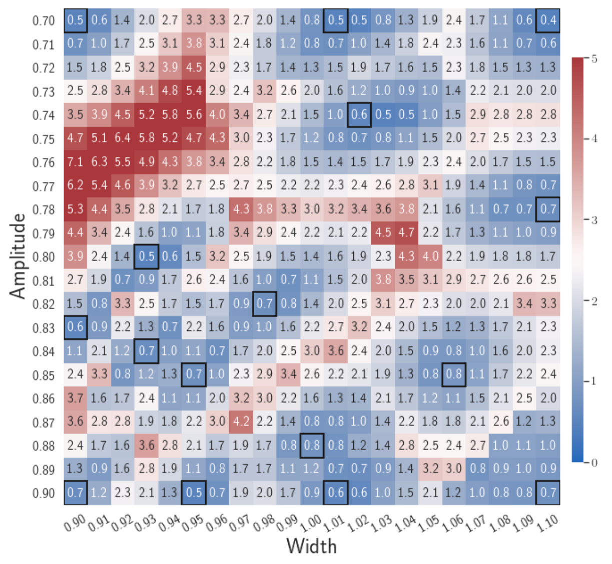
Figure 8.7. The history of the loss function and the maximum residual-based error of the sampled parameter for 1D Burgers problem. The KNN parameter $k = 4$ is used for greedy sampling procedure during training.

results are presented.

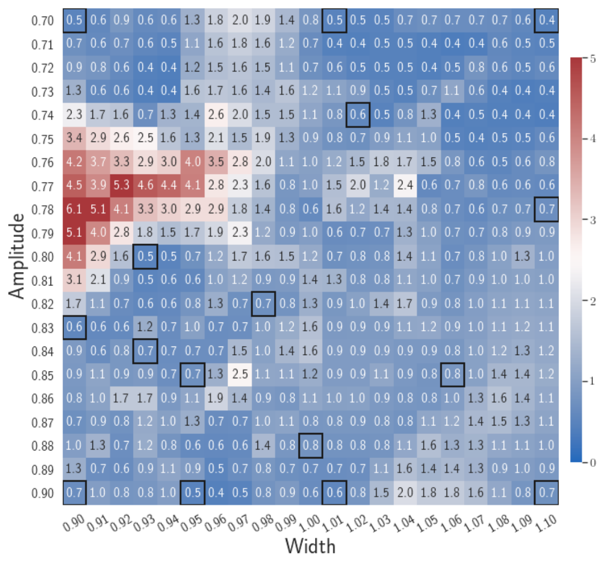
Case 2: gLaSDI vs. LaSDI

In the second test, the autoencoder with an architecture of 1,001-100-5 and linear DI models are considered. The same parameter space with 21×21 parameter cases in total is considered. The gLaSDI training with adaptive greedy sampling is performed until the total number of sampled parameter points reaches 25. To investigate the effects of adaptive greedy sampling on model performances, a LaSDI model with the same architecture of the autoencoder and DI models is trained using 25 predefined training points uniformly distributed in a 5×5 grid in the parameter space. The performance of gLaSDI and LaSDI are compared and discussed.

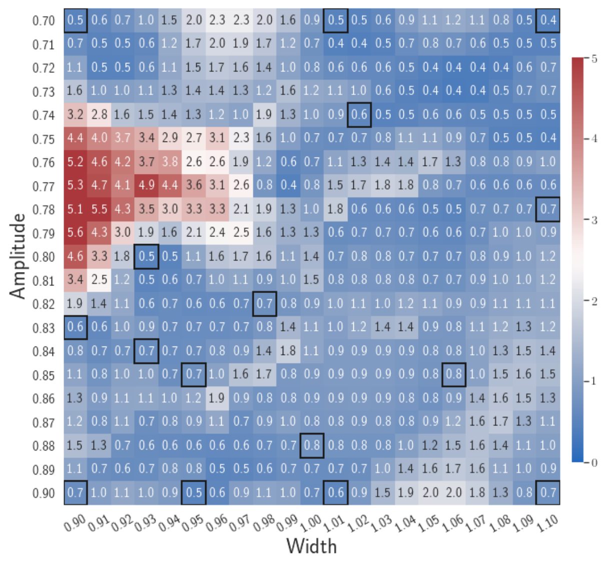
Fig. 8.9(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. The latent-space dynamics from gLaSDI is more linear and simpler than that from LaSDI. gLaSDI also achieves a better agreement between the encoder and the DI predictions, which is attributed by the interactive learning of gLaSDI. Note that the sequential training of the autoencoder and the DI models of the LaSDI could lead to more complicated and nonlinear latent-space dynamics because of the lack of interaction between the



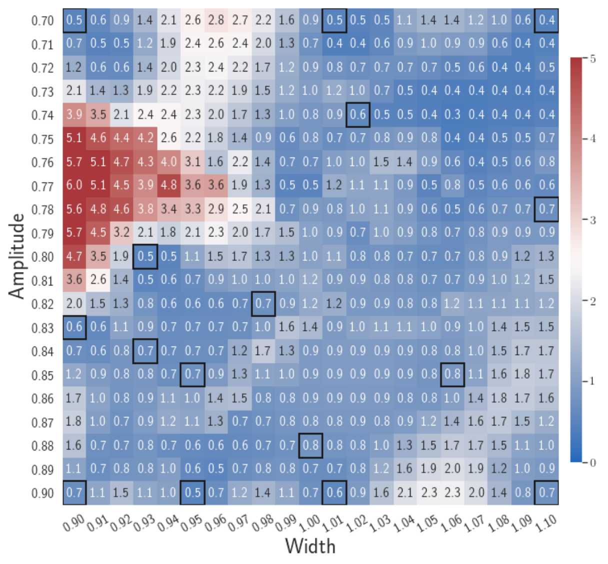
(a) $k = 1$ evaluation



(b) $k = 3$ evaluation



(c) $k = 4$ evaluation



(d) $k = 5$ evaluation

Figure 8.8. Maximum relative errors in the parameter space $\mathscr{D}^h$ evaluated with different values of k for KNN convex interpolation of the DI coefficient matrices of the testing parameter for 1D Burgers problem: (a) $k = 1$, (b) $k = 3$, (c) $k = 4$, (d) $k = 5$. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the parameter points sampled from training. $k = 4$ is used for greedy sampling procedure during training.

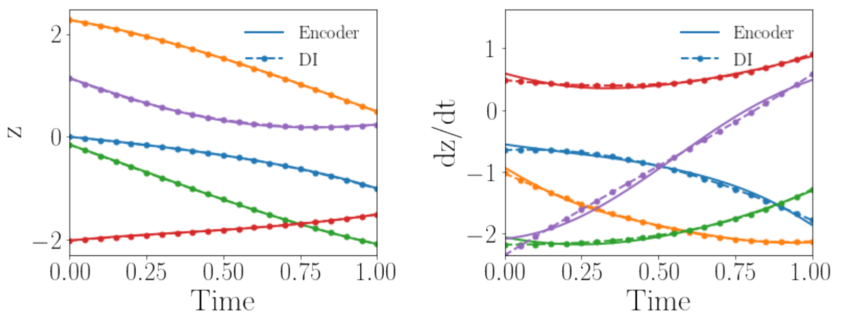
latent-space dynamics learned by the autoencoder and the DI models. As a consequence, it could pose challenges for the subsequent training of DI models to capture the latent-space dynamics learned by the autoencoder and cause higher prediction errors.

Fig. 8.9(c-d) show the maximum relative errors of LaSDI and gLaSDI predictions in the prescribed parameter space, respectively, where the black square boxes indicate the locations of the sampled parameter points. For LaSDI, the training parameter points are pre-selected and the associated high-fidelity solutions are obtained before training starts, whereas for gLaSDI, the training parameter points are sampled adaptively on-the-fly during the training, within which the greedy sampling algorithm is combined with the physics-informed residual-based error indicator, as introduced in Section 8.3.4, which allows the points with the maximum error to be selected. Thus gLaSDI enhances the accuracy with less number of sampled parameter points than LaSDI. It can be observed that gLaSDI tends to have denser sampling in the lower range of the parameter space. Fig. 8.9(d) shows that gLaSDI achieves the maximum relative error of 1.9% in the whole parameter space, which is much lower than 4.5% of LaSDI in Fig. 8.9(c).

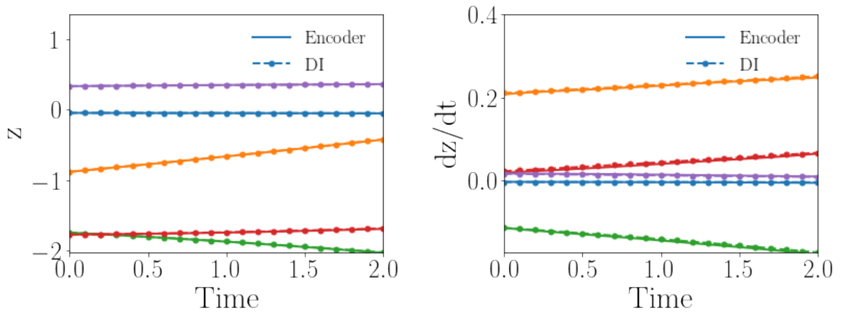
Case 3: Effects of the loss terms

As mentioned in Sections 8.3.1 and 8.3.2, there are three distinct terms in the loss function of the gLaSDI training. Each term includes a different set of trainable parameters. For example, the reconstruction term $\mathscr{L}_{recon}$ includes θ_{enc} and θ_{dec}; the latent-space dynamics identification term $\mathscr{L}_{\dot{\mathbf{z}}}$ includes θ_{enc} and $\{\Xi^{(i)}\}_{i\in\mathbb{N}(N_\mu)}$; and $\mathscr{L}_{\dot{\mathbf{u}}}$ includes all the trainable parameters, i.e., θ_{enc}, θ_{dec}, and $\{\Xi^{(i)}\}_{i\in\mathbb{N}(N_\mu)}$. Therefore, in this section, we demonstrate the effects of these loss terms.

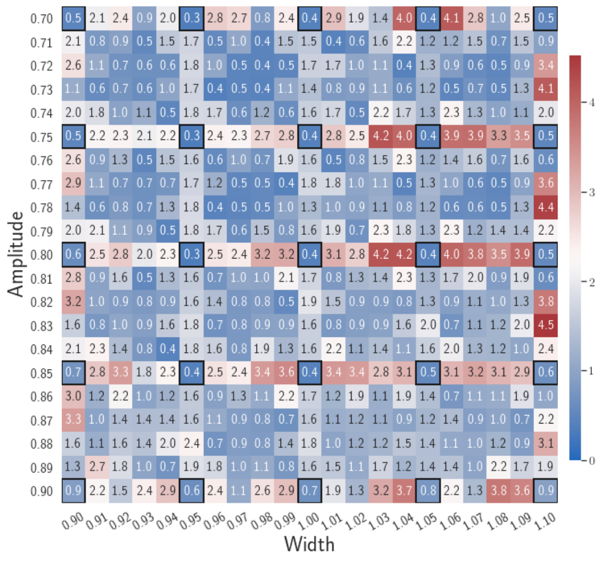
Given the same settings of the gLaSDI model and the parameter space considered in Section 8.4.1, the gLaSDI is trained by only one loss term, $\mathscr{L}_{\dot{\mathbf{u}}}$ (Eq. (8.13)), that involves all the trainable parameters. The predicted latent-space dynamics and maximum relative errors in the parameter space are shown in Fig. 8.10(a) and (c), respectively. The large deviation in the predicted latent-space dynamics between the encoder and the DI model leads to large prediction errors in the physical dynamics. It is also observed that the identified latent-space



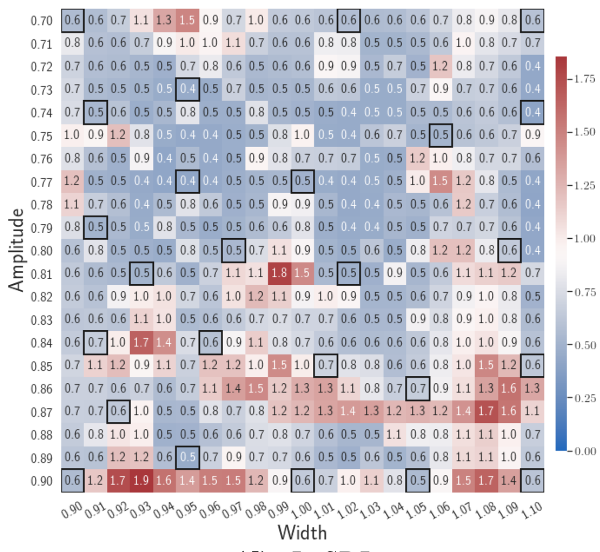
(a) LaSDI



(b) gLaSDI



(c) LaSDI



(d) gLaSDI

Figure 8.9. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (1,001-100-5) and dynamics identification models (linear) for 1D Burgers problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 4$ and (d) gLaSDI with $k = 3$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled training parameter points. The KNN parameter $k = 1$ is used for greedy sampling procedure during training of gLaSDI.

dynamics is more nonlinear, compared with that shown in Fig. 8.9(b). It implies that $\mathscr{L}_{\dot{\mathbf{u}}}$ cannot impose sufficient constraints on the trainable parameters to identify simple latent-space dynamics although it includes all the trainable parameters.

In the second test, the gLaSDI is trained by $\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}}$ with different values of the regularization parameter β_1. The motivation is to see if we can achieve as good accuracy with only first two loss terms as the one with all three loss terms. The maximum relative errors in the parameter space corresponding to different values of β_1 are summarized in Table 8.1. Using $\beta_1 = 10^{-2}$ yields the best model. Compared with the latent-space dynamics of gLaSDI trained with all three loss terms, as shown in Fig. 8.9(b), the gLaSDI trained without $\mathscr{L}_{\dot{\mathbf{u}}}$ produces relatively more nonlinear latent-space dynamics, as shown in Fig. 8.10(b). The comparison also shows that the constraint imposed by $\mathscr{L}_{\dot{\mathbf{u}}}$ on model training enhances the generalization performance of gLaSDI, reducing the maximum relative error in the whole parameter space from 4.5% to 1.9%, as shown in Fig. 8.9(d) and 8.10(d).

Table 8.1. The maximum relative errors in the parameter space of the gLaSDI trained by $\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}}$ with different values of β_1.

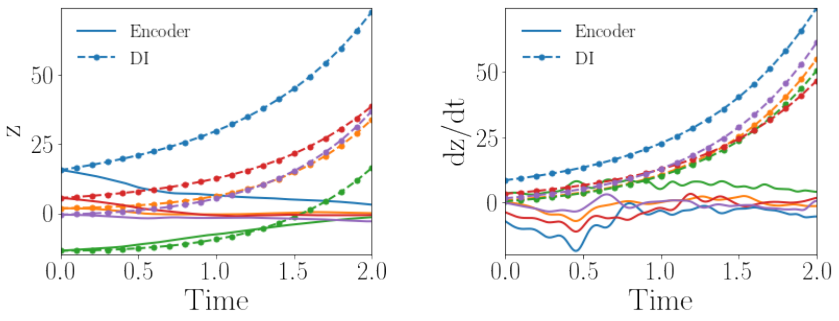
β_1	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^1	10^2	10^3
e^{max} (%)	11.5	8.5	4.5	4.7	4.6	14.5	20.9	23.8

8.4.2 2D Burgers equation

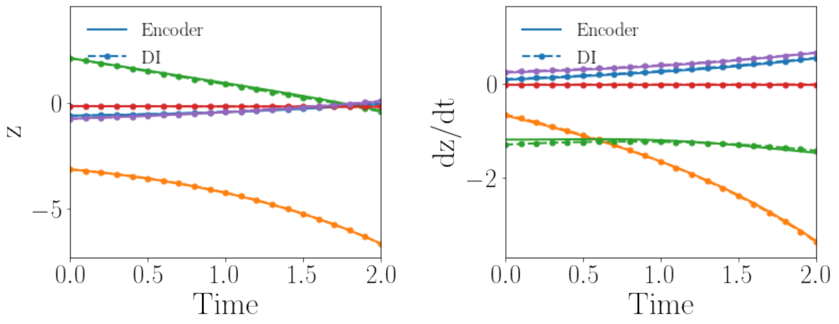
A 2D parameterized inviscid Burgers equation is considered

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{1}{Re} \Delta \mathbf{u}, \quad \mathbf{x} \in \Omega = [-3,3] \times [-3,3], \quad t \in [0,1], \tag{8.28a}$$

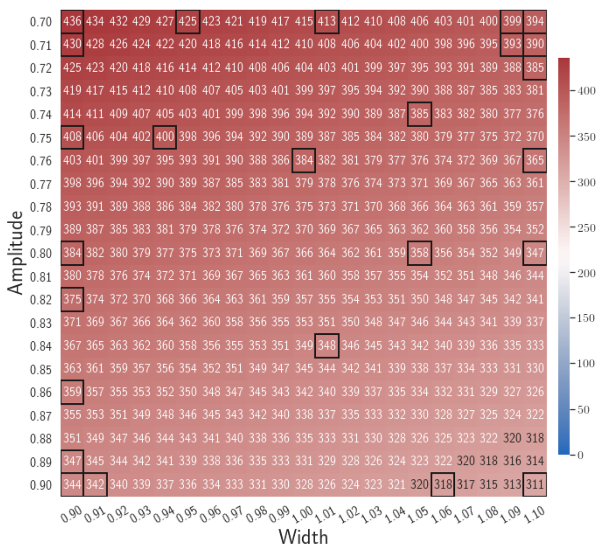
$$\mathbf{u}(\mathbf{x}, t; \mu) = \mathbf{0} \quad \text{on} \quad \partial\Omega. \tag{8.28b}$$



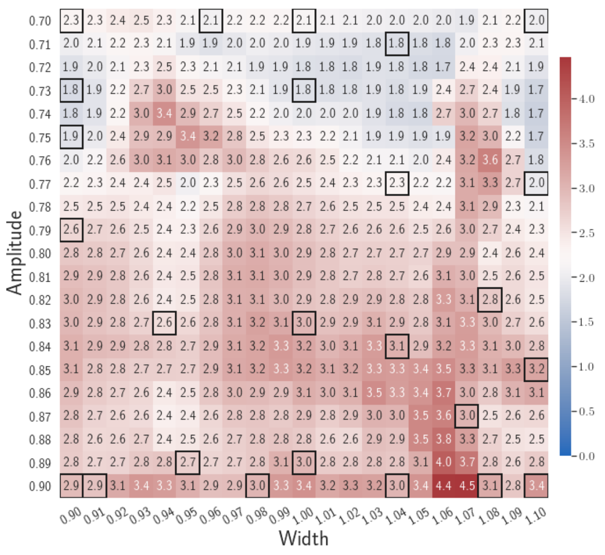
(a) $\mathscr{L} = \mathscr{L}_{\dot{\mathbf{u}}}$



(b) $\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}}$



(c) $\mathscr{L} = \mathscr{L}_{\dot{\mathbf{u}}}$



(d) $\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}}$

Figure 8.10. Results of gLaSDI trained with different loss terms for 1D Burgers problem. An autoencoder of (1,001-100-5) and linear dynamics identification models are used. The latent dynamics predicted by the encoder and the dynamics identification model from the gLaSDI trained by (a) $\mathscr{L} = \mathscr{L}_{\dot{\mathbf{u}}}$ and (b) $\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}}$, with $\beta_1 = 10^{-2}$. The maximum relative errors (evaluated with $k = 4$) in the parameter space $\mathscr{D}^h$ from the gLaSDI trained by (c) $\mathscr{L} = \mathscr{L}_{\dot{\mathbf{u}}}$ and (d) $\mathscr{L} = \mathscr{L}_{recon} + \beta_1 \mathscr{L}_{\dot{\mathbf{z}}}$, with $\beta_1 = 10^{-2}$. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled training parameter points. The KNN parameter $k = 1$ is used for greedy sampling procedure during training of gLaSDI.

Eq. (8.28b) is an essential boundary condition. The initial condition is parameterized by the amplitude a and the width w, defined as

$$\mathbf{u}(\mathbf{x},0;\mu) = ae^{-\frac{||\mathbf{x}||^2}{w^2}}, \tag{8.29}$$

where $\mu = \{a,w\}$. A uniform spatial discretization with 60×60 discrete points is applied. The first order spatial derivative is approximated by the backward difference scheme, while the diffusion term is approximated by the central difference scheme. The semi-discertized system is solved by using the implicit backward Euler time integrator with a uniform time step of $\Delta t = 1/200$ to obtain the full-order model solutions. The solution fields of the first velocity component at different time steps for the parameter case $(a = 0.7, w = 0.9)$ are shown in Fig. 8.11.

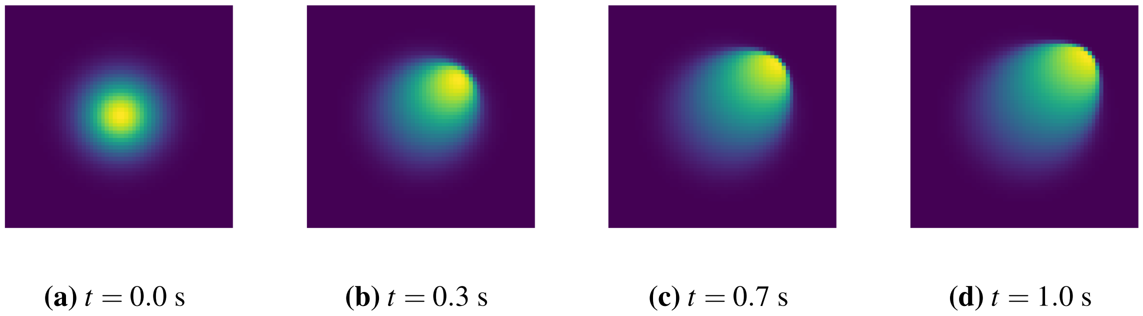

(a) $t = 0.0$ s **(b)** $t = 0.3$ s **(c)** $t = 0.7$ s **(d)** $t = 1.0$ s

Figure 8.11. The solution fields of the first velocity component at different time steps for the parameter case $(a = 0.7, w = 0.9)$: (a) $t = 0.0$ s, (b) $t = 0.3$ s, (c) $t = 0.7$ s, (d) $t = 1.0$ s for 2D Burgers problem.

Case 1: Comparison between gLaSDI and LaSDI

In the first test, a discrete parameter space $\mathcal{D}^h$ is constituted by the parameters of the initial condition, including the width, $w \in [0.9, 1.1]$, and the amplitude, $a \in [0.7, 0.9]$, each with 21 evenly distributed discrete points in the respective parameter range. The autoencoder with an architecture of 7,200-100-5 and quadratic DI models are considered. The gLaSDI training is performed until the total number of sampled parameter points reaches 36. A LaSDI model with the same architecture of the autoencoder and DI models is trained using 36 predefined training points uniformly distributed in a 6×6 grid in the parameter space. The performance of gLaSDI and LaSDI are compared and discussed.

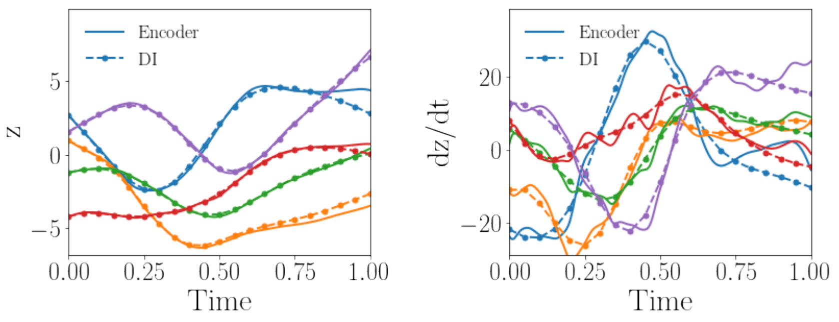
Fig. 8.12(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. Again, the latent-space dynamics of gLaSDI is simpler than that of LaSDI, with a better agreement between the encoder and the DI predictions, which is attributed by the interactive learning of gLaSDI.

Fig. 8.12(c-d) show the maximum relative error of LaSDI and gLaSDI predictions in the prescribed parameter space, respectively. The gLaSDI achieves the maximum relative error of 5% in the whole parameter space, much lower than 255% of LaSDI. The poor accuracy of LaSDI could be caused by the deviation between the DI predicted dynamics and the encoder predicted dynamics. It is also observed that gLaSDI tends to have denser sampling in the lower range of the parameter space. This demonstrates the importance of the physics-informed greedy sampling procedure. Compared with the high-fidelity simulation based on an in-house Python code, the gLaSDI model achieves 1,740$\times$ speed-up.

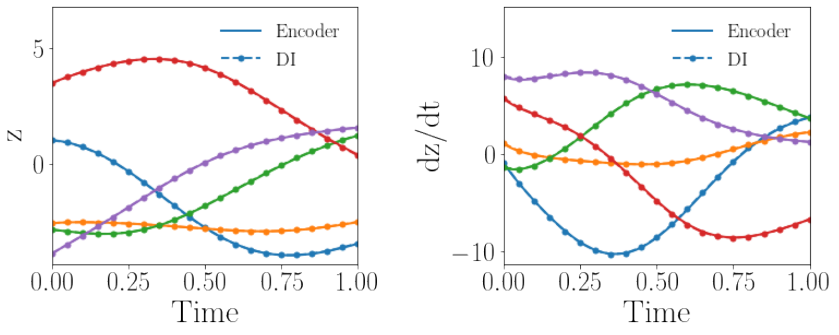
Case 2: Effects of the polynomial order in DI models and latent space dimension

In the second test, we want to see if simpler latent-space dynamics can be achieved by gLaSDI and how it affects the reduced-order modeling accuracy. The same parameter space with 21×21 parameter cases in total is considered. The latent dimension is reduced from 5 to 3 and the polynomial order of the DI models is reduced from quadratic to linear. The autoencoder architecture becomes 7,200-100-3. The gLaSDI training is performed until the total number of sampled parameter porints reaches 36. A LaSDI model with the same architecture of the autoencoder and DI models is trained using 36 predefined training parameter points uniformly distributed in a 6×6 grid in the parameter space.

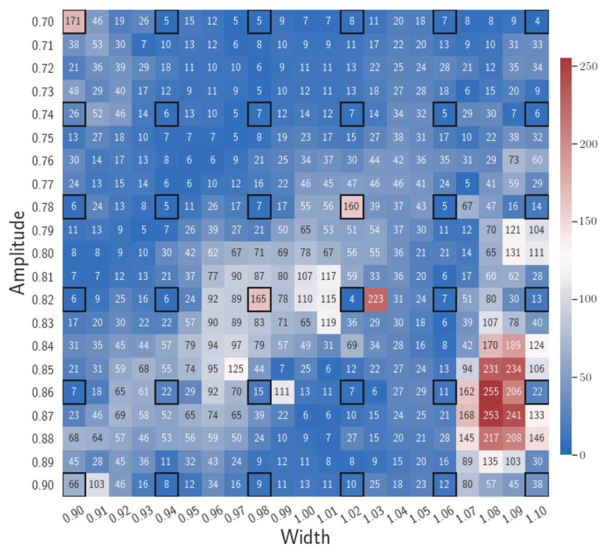
Fig. 8.13(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. The latent-space dynamics of gLaSDI is



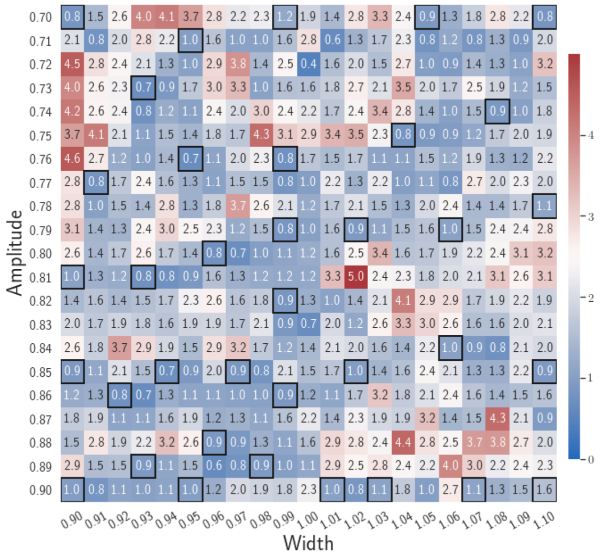
(a) LaSDI



(b) gLaSDI



(c) LaSDI



(d) gLaSDI

Figure 8.12. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (7,200-100-5) and dynamics identification models (**quadratic**) for 2D Burgers problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 4$ and (d) gLaSDI with $k = 3$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the location of the sampled training points. The KNN parameter $k = 1$ is used for greedy sampling procedure for the training of gLaSDI.

simpler than that of LaSDI, with a better agreement between the encoder and the DI predictions. It also demonstrates that gLaSDI could learn simpler latent-space dynamics, which enhances the reduced-order modeling efficiency.Compared with the high-fidelity simulation based on an in-house Python code, the gLaSDI model achieves 4,417× speed-up, which is 2.54 times of the speed-up achieved by the gLaSDI model that has a latent dimension of 5 and quadratic DI models, as shown in Section 8.4.2.

Fig. 8.13(c-d) show the maximum relative error of LaSDI and gLaSDI predictions on each parameter case in the prescribed parameter space, respectively. Compared with the example with a latent dimension of five and quadratic DI models, as shown in the previous subsection, both gLaSDI and LaSDI achieve lower maximum relative errors, reduced from 5.0% to 4.6% and from 255% to 22%, respectively. It indicates that reducing the complexity of the latent-space dynamics allows the DI models of LaSDI to capture the encoder predicted dynamics more accurately although the error level of LaSDI is still larger than that of gLaSDI due to no interactions between the autoencoder and DI models during training. Simplifying latent-space dynamics results in higher reduced-order modeling accuracy of both LaSDI and gLaSDI for this example. We speculate that the intrinsic latent space dimension for the 2D Burgers problem is close to three.

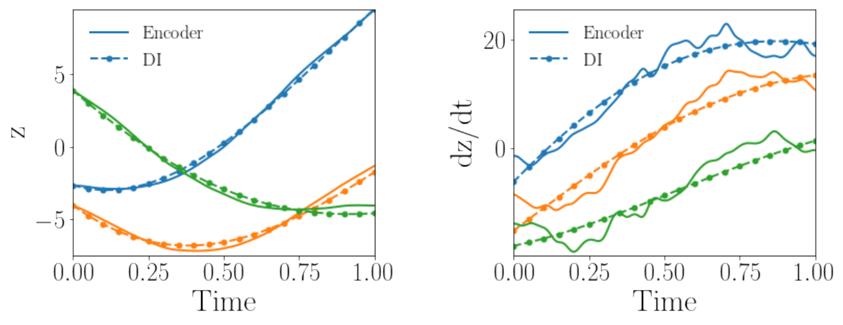
8.4.3 Nonlinear time-dependent heat conduction

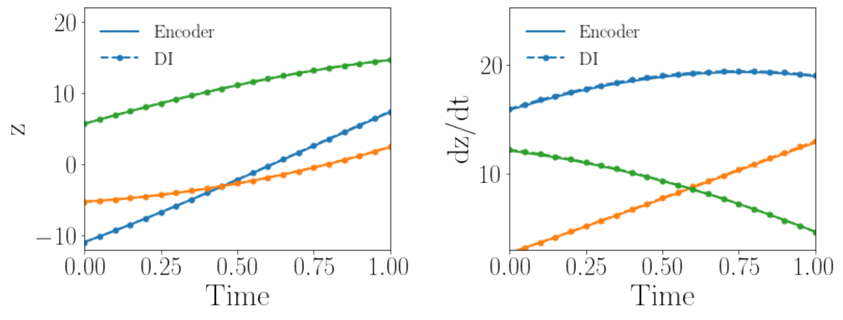
A 2D parameterized nonlinear time-dependent heat conduction problem is considered

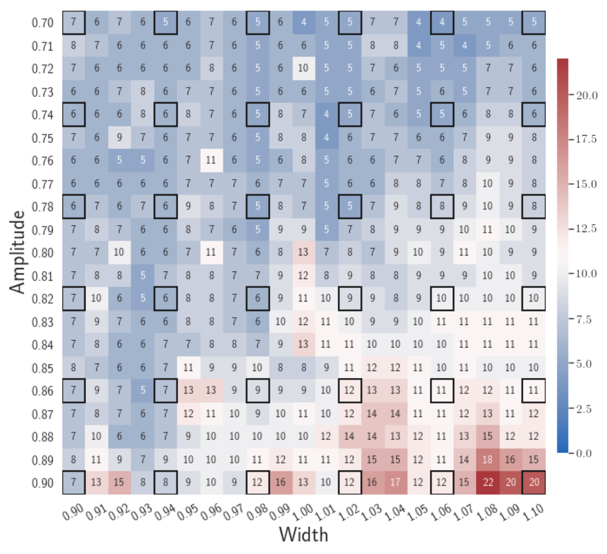
$$\frac{\partial u}{\partial t} = \nabla \cdot (\kappa + \alpha u)\nabla u \in \Omega = [0,1] \times [0,1], \quad t \in [0,0.3], \tag{8.30a}$$

$$\frac{\partial u}{\partial \mathbf{n}} = \mathbf{0} \quad \text{on} \quad \partial\Omega, \tag{8.30b}$$

where **n** denotes the unit normal vector. Eq. (8.30b) is a natural insulating boundary condition. The coefficients $\kappa = 0.5$ and $\alpha = 0.1$ are adopted in the following examples. The initial condition


(a) LaSDI


(b) gLaSDI


(c) LaSDI

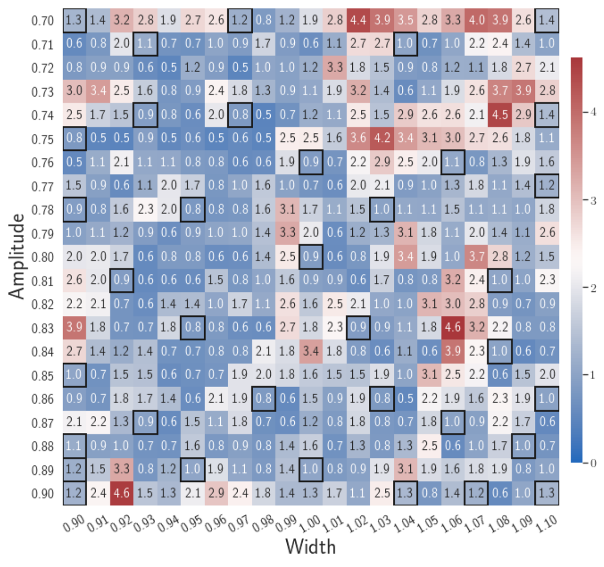
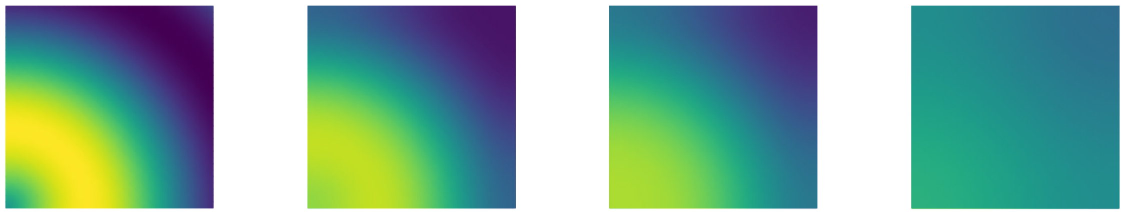

(d) gLaSDI

Figure 8.13. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (7,200-100-3) and dynamics identification models (**linear**) for 2D Burgers problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 3$ and (d) gLaSDI with $k = 3$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the location of the sampled training points. The KNN parameter $k = 1$ is used for greedy sampling procedure during training of gLaSDI.

is parameterized as

$$u(\mathbf{x},0;\mu) = a\sin(w||\mathbf{x}||_{L_2}) + a, \tag{8.31}$$

where $\mu = \{a, w\}$ denotes the paraemeters of the initial condition. The spatial domain is discretized by first-order square finite elements constructed on a uniform grid of 33×33 discrete points. The implicit backward Euler time integrator with a uniform time step of $\Delta t = 0.005$ is employed. The conductivity coefficient is computed by linearizing the problem with the temperature field from the previous time step. The solution fields at different time steps for the parameter case $(w = 4, a = 1)$ are shown in Fig. 8.14.



(a) $t = 0.0$ s **(b)** $t = 0.03$ s **(c)** $t = 0.07$ s **(d)** $t = 0.3$ s

Figure 8.14. The solution fields at different time steps for the parameter case $(w = 4, a = 1)$: (a) $t = 0.0$ s, (b) $t = 0.03$ s, (c) $t = 0.07$ s, (d) $t = 0.3$ s for the nonlinear time-dependent heat conduction problem.

Case 1: Comparison between gLaSDI and LaSDI

In the first test, a parameter space $\mathscr{D}^h$ is constituted by the parameters of the initial condition, including the $w \in [4.0, 4.3]$ and $a \in [1.0, 1.4]$, each with 21 evenly distributed discrete points in the respective parameter range. The autoencoder with an architecture of 1,089-100-3 and quadratic DI models are considered. The gLaSDI training is performed until the total number of sampled parameter points reaches 25. A LaSDI model with the same architecture of the autoencoder and DI models is trained using 25 predefined training parameter points uniformly distributed in a 5×5 grid in the parameter space. The performances of gLaSDI and LaSDI are compared and discussed.

Fig. 8.15(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. Again, gLaSDI achieves a better agreement between the encoder and the DI prediction than the ones for LaSDI.

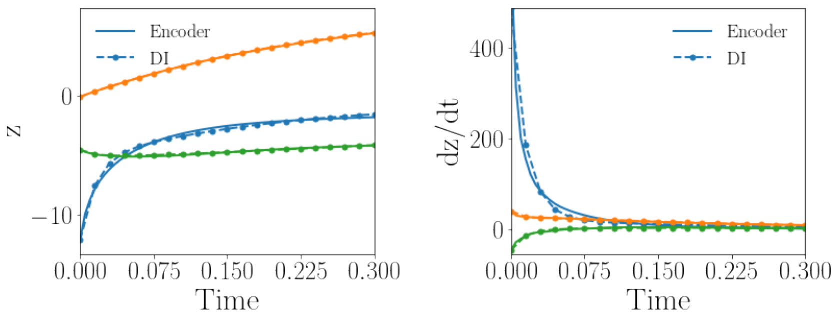
Fig. 8.15(c-d) show the maximum relative error of LaSDI and gLaSDI predictions in the prescribed parameter space, respectively. The gLaSDI achieves higher prediction accuracy than the LaSDI with the maximum relative error of 3.1% in the whole parameter space, compared to 7.1% of LaSDI. Compared with the high-fidelity simulation based on MFEM [220], the gLaSDI model achieves 66$\times$ speed-up.

Case 2: Effects of the polynomial order in DI models

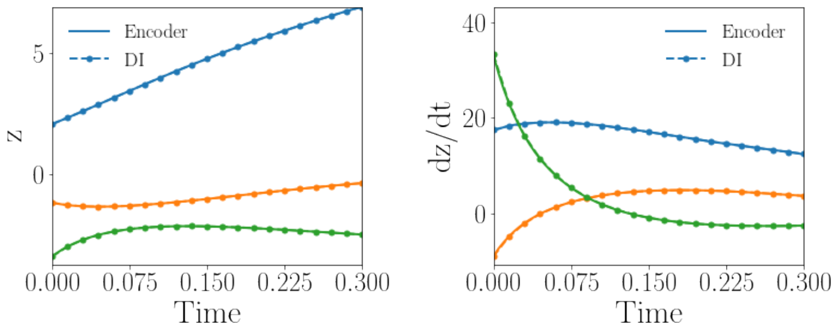
In the second test, we want to see if simpler latent-space dynamics can be achieved by gLaSDI and how it affects the reduced-order modeling accuracy. The same settings as the previous example are considered, including the parameter space, the autoencoder architecture (1,089-100-3), the number of training points (25), except that the polynomial order of the DI models is reduced from quadratic to linear.

Fig. 8.16(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. Compared with the LaSDI's latent-space dynamics in the previous example, as shown in Fig. 8.15(a-b), the agreement between the encoder and the DI predictions in this example improves, although not as good as that of gLaSDI. The dynamics learned by gLaSDI is simpler than the previous example with quadratic DI models. Compared with the high-fidelity simulation based on MFEM [220], the gLaSDI model achieves 205$\times$ speed-up, which is 3.11 times of the speed-up achieved by the gLaSDI model that has quadratic DI models, as shown in Section 8.4.3. It further demonstrates that gLaSDI allows learning simpler latent-space dynamics, which could enhance the reduced-order modeling efficiency.

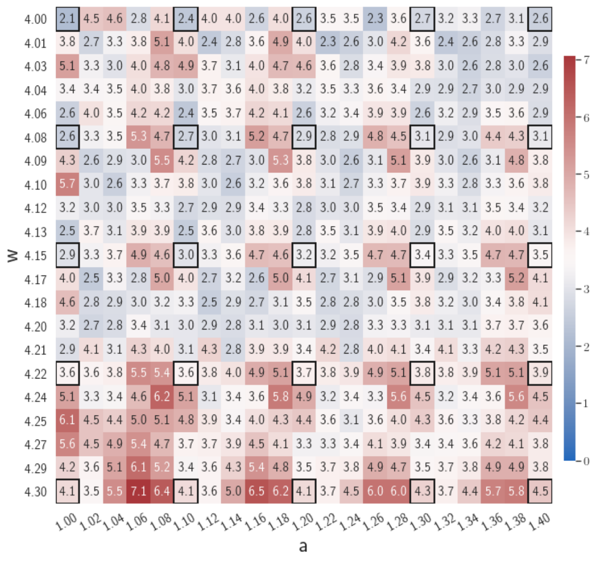
Fig. 8.16(c-d) show the maximum relative error of LaSDI and gLaSDI predictions in the prescribed parameter space, respectively. Compared with the example with quadratic DI models, as shown in the previous subsection, the maximum relative error achieved by gLaSDI is reduced



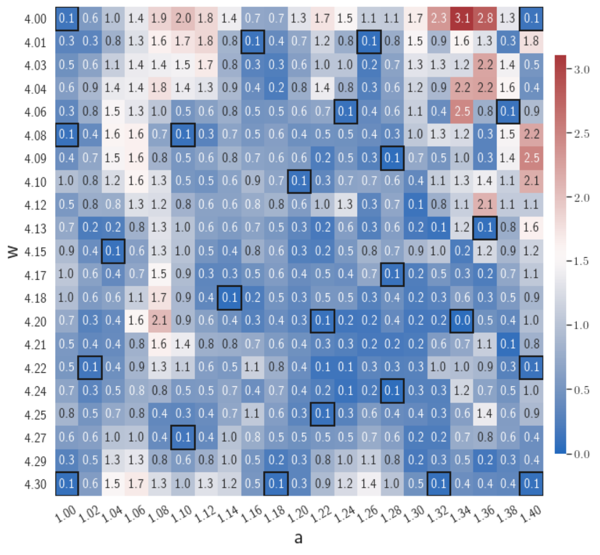
(a) LaSDI



(b) gLaSDI



(c) LaSDI



(d) gLaSDI

Figure 8.15. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (1,089-100-3) and dynamics identification models (**quadratic**) for the nonlinear time dependent heat conduction problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 4$ and (d) gLaSDI with $k = 3$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled training points. The KNN parameter $k = 1$ is used for greedy sampling procedure during training of gLaSDI.

from 3.1% to 1.4%, while that achieved by LaSDI is reduced from 7.1% to 5.7%. Simplifying latent-space dynamics contributes to higher reduced-order modeling accuracy of both LaSDI and gLaSDI in this example. This implies that the higher order in DI model does not always help improving the accuracy.

8.4.4 Time-dependent Radial advection

A 2D parameterized time-dependent radial advection problem is considered

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{u} = \in \Omega = [-1,1] \times [-1,1], \quad t \in [0,3], \tag{8.32a}$$

$$u(\mathbf{x},t;\mu) = \mathbf{0} \quad \text{on} \quad \partial\Omega, \tag{8.32b}$$

where Eq. (8.32b) is a boundary condition and **v** denotes the fluid velocity, defined as

$$\mathbf{v} = \frac{\pi}{2} d[x_2, -x_1]^T, \tag{8.33}$$

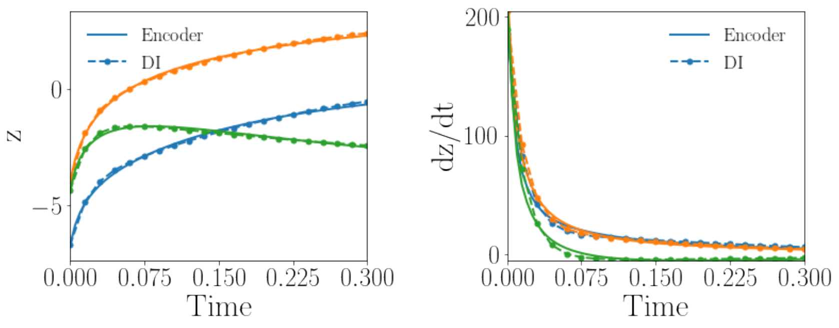
with $d = (1-x_1^2)^2(1-x_2^2)^2$. The initial condition is defined as

$$u(\mathbf{x},0;\mu) = \sin(w_1 x_1)\sin(w_2 x_2), \tag{8.34}$$

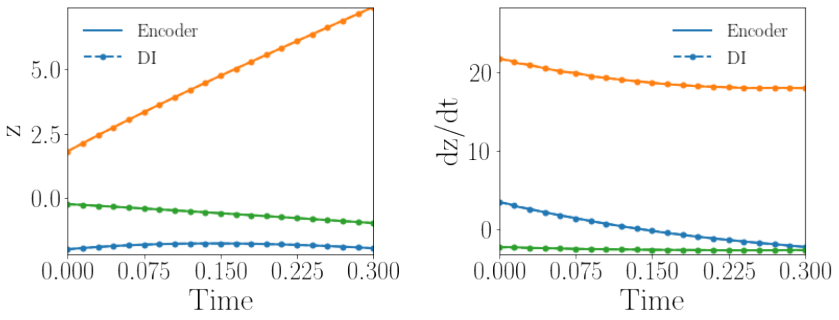
where $\mu = \{w_1, w_2\}$ denotes the paraemeters of the initial condition. The spatial domain is discretized by first-order periodic square finite elements constructed on a uniform grid of 96×96 discrete points. The fourth-order Runge-Kutta explicit time integrator with a uniform time step of $\Delta t = 0.01$ is employed. The solution fields at different time steps for the parameter case $(w_1 = 1.5, w_2 = 2.0)$ are shown in Fig. 8.17.

Case 1: Comparison between gLaSDI and LaSDI

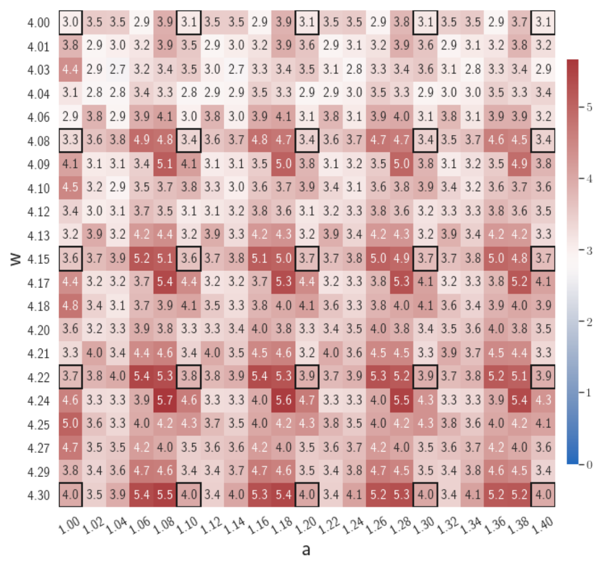
In the first test, a parameter space $\mathcal{D}^h$ is constituted by the parameters of the initial condition, including the $w_1 \in [1.5, 1.8]$ and $w_2 \in [2.0, 2.3]$, each with 21 evenly distributed



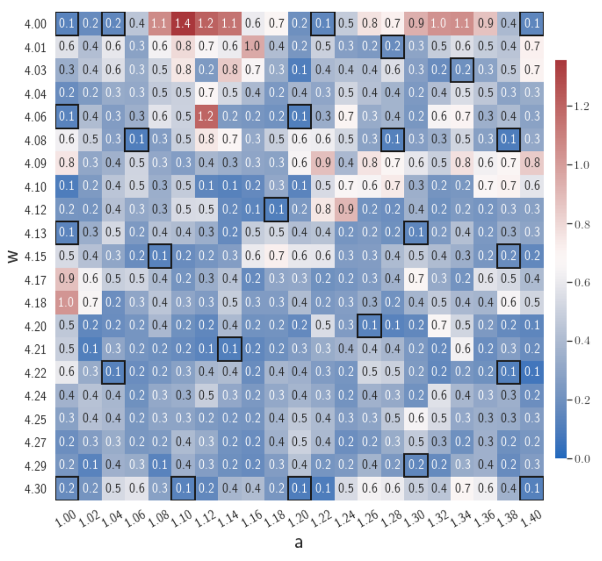
(a) LaSDI



(b) gLaSDI

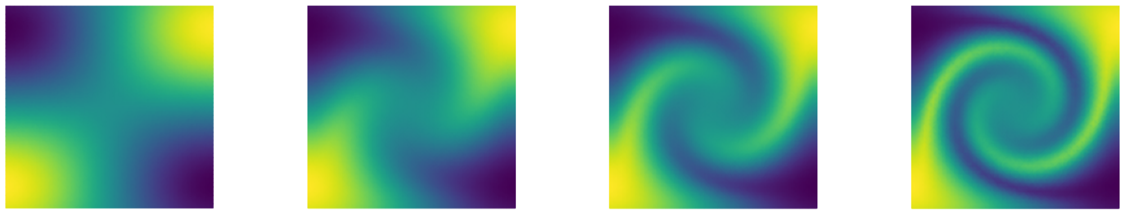


(c) LaSDI



(d) gLaSDI

Figure 8.16. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (1,089-100-3) and dynamics identification models (**linear**) for the nonlinear time dependent heat conduction problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 4$ and (d) gLaSDI with $k = 4$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled training points. The KNN parameter, $k = 1$ is used for greedy sampling procedure during training of gLaSDI.



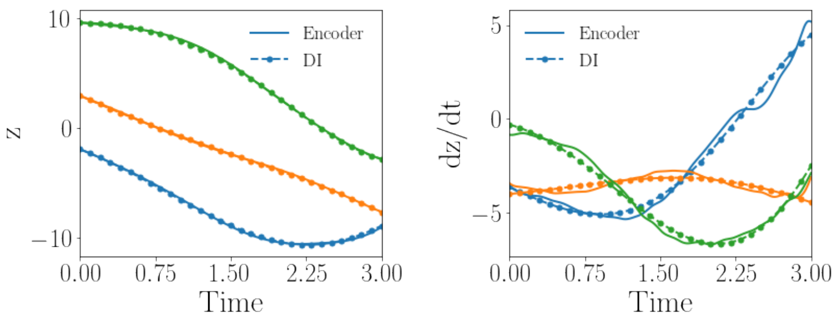
(a) $t = 0.0$ s **(b)** $t = 0.7$ s **(c)** $t = 1.7$ s **(d)** $t = 3.0$ s

Figure 8.17. The solution fields at different time steps for the parameter case $(w_1 = 1.5, w_2 = 2.0)$: (a) $t = 0.0$ s, (b) $t = 0.7$ s, (c) $t = 1.7$ s, (d) $t = 3.0$ s.

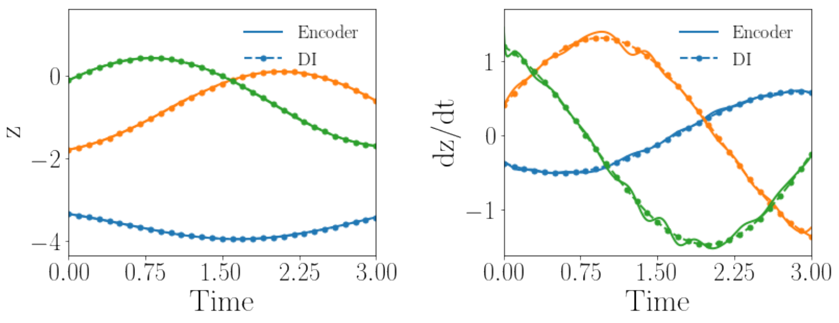
discrete points in the respective parameter range. The autoencoder with an architecture of 9,216-100-3 and linear DI models are considered. The gLaSDI training is performed until the total number of sampled parameter points reaches 25. A LaSDI model with the same architecture of the autoencoder and DI models is trained using 25 predefined training points uniformly distributed in a 5×5 grid in the parameter space. The performances of gLaSDI and LaSDI are compared and discussed.

Fig. 8.18(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. The gLaSDI achieves simpler time derivative latent-space dynamics than LaSDI, with a better agreement between the encoder and the DI prediction.

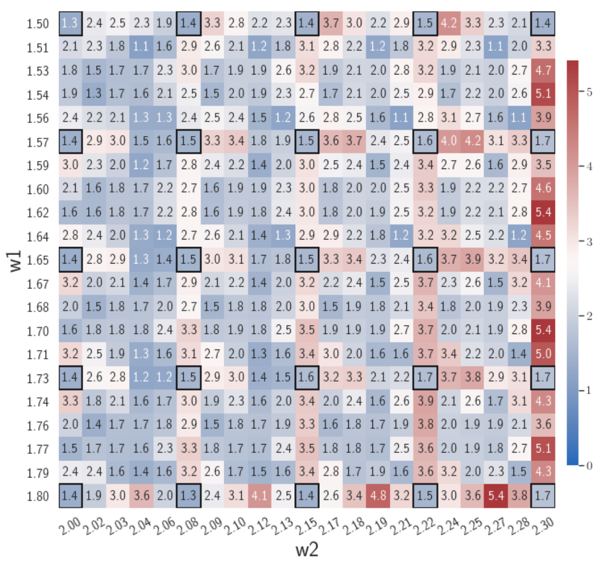
Fig. 8.18(c-d) show the maximum relative error of LaSDI and gLaSDI predictions in the prescribed parameter space, respectively. The gLaSDI achieves higher prediction accuracy than the LaSDI with the maximum relative error of 2.0% in the whole parameter space, compared to 5.4% of LaSDI. It is observed that gLaSDI tends to have denser sampling in the regime with higher parameter values, concentrating at the bottom-right corner of the parameter space, which implies that more vibrant change in dynamics is present for higher parameter values, requiring more local DI models there. Compared with the high-fidelity simulation based on MFEM [220], the gLaSDI model achieves 200× speed-up.



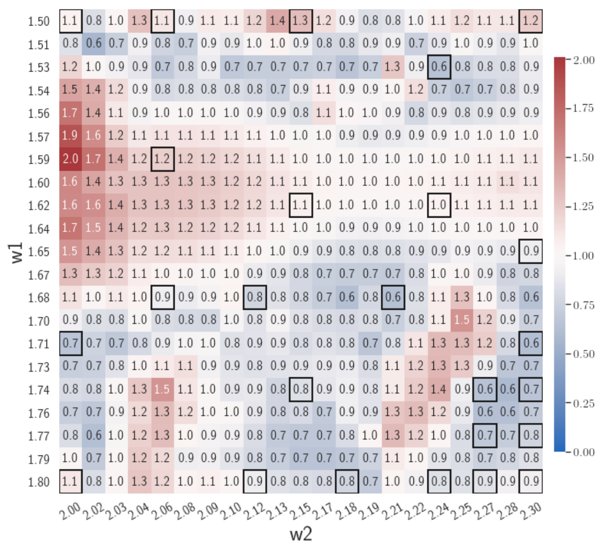
(a) LaSDI



(b) gLaSDI



(c) LaSDI



(d) gLaSDI

Figure 8.18. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (9,216-100-3) and dynamics identification models (**linear**) for the time dependent radial advection problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 4$ and (d) gLaSDI with $k = 4$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled training points. The KNN parameter $k = 1$ is used for greedy sampling procedure during training of gLaSDI.

Case 2: Effects of the size of parameter space

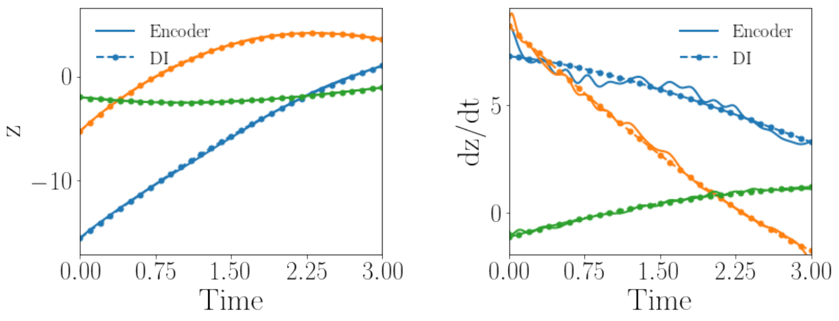
In the second test, we want to see how the size of parameter space affect the model performances of LaSDI and gLaSDI. A larger parameter space $\mathscr{D}^h$ is considered and constituted by the parameters of the initial condition, including the $w_1 \in [1.5, 2.0]$ and $w_2 \in [2.0, 2.5]$, each with 21 evenly distributed discrete points in the respective parameter range. Other settings remain the same as those used in the previous example, including the number of training points set 25.

Fig. 8.19(a-b) show the latent-space dynamics predicted by the trained encoder and the DI model from LaSDI and gLaSDI, respectively. It again shows that gLaSDI learns smoother latent-space dynamics than LaSDI, with a better agreement between the encoder and the DI predictions than LaSDI.

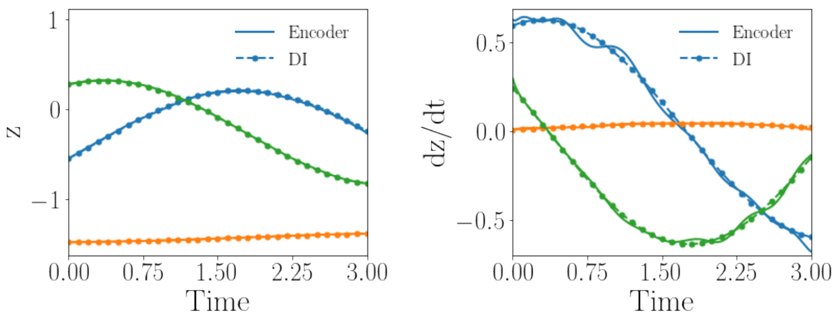
Fig. 8.19(c-d) show the maximum relative error of LaSDI and gLaSDI predictions in the prescribed parameter space, respectively. Compared with the example with a smaller parameter space, as shown in the previous subsection, the maximum relative error achieved by gLaSDI in the whole parameter space increases from 2.0% to 3.3%, while that achieved by LaSDI increases from 5.4% to 24%. It shows that gLaSDI maintains high accuracy even when the parameter space is enlarged, while LaSDI's error increases significantly due to non-optimal sampling and the mismatch between the encoder and DI predictions. It is interesting to note that changing the parameter space affects the distribution of gLaSDI sampling.

8.5 Summary

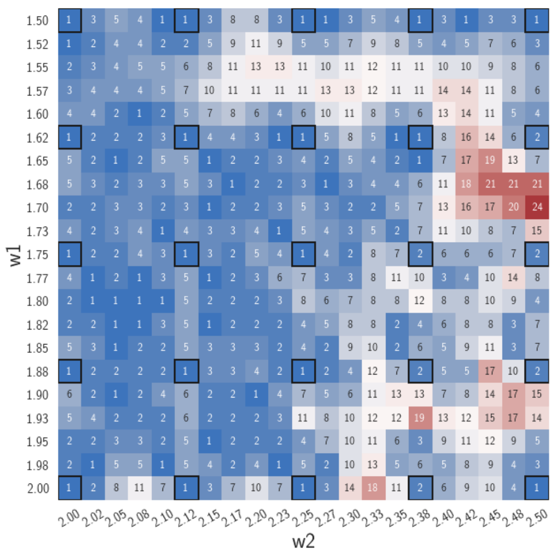
In this study, we introduced a physics-informed greedy parametric latent-space dynamics identification (gLaSDI) framework for accurate, efficient, and robust data-driven computing of high-dimensional nonlinear dynamical systems. The proposed gLaSDI framework is composed of an autoencoder that performs nonlinear compression of high-dimensional data and discovers intrinsic latent representations as well as dynamics identification (DI) models that capture local latent-space dynamics. The autoencoder and DI models are trained interactively and



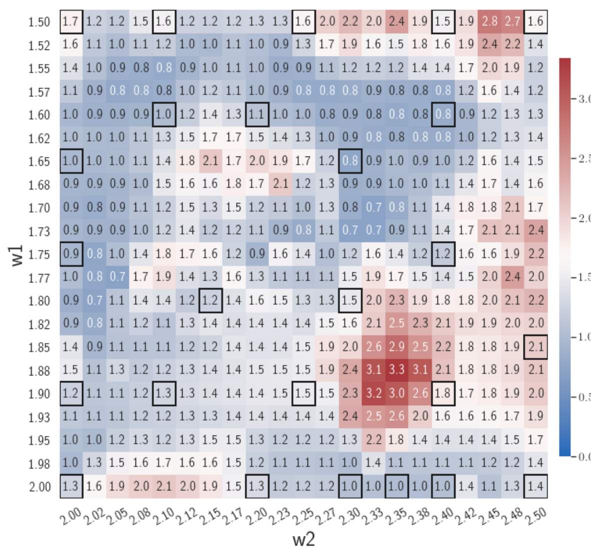
(a) LaSDI



(b) gLaSDI



(c) LaSDI



(d) gLaSDI

Figure 8.19. Comparison between LaSDI and gLaSDI with the same architecture of autoencoder (9,216-100-3) and dynamics identification models (**linear**) for the time dependent radial advection problem. The latent dynamics predicted by the trained encoder and the trained dynamics identification model from (a) LaSDI and (b) gLaSDI. The maximum relative errors in the parameter space $\mathscr{D}^h$ from (c) LaSDI with $k = 4$ and (d) gLaSDI with $k = 4$ for KNN convex interpolation during evaluation. The number on each box denotes the maximum relative error of the associated parameter case. The black square boxes indicate the locations of the sampled training points. The KNN parameter $k = 1$ is used for greedy sampling procedure during training of gLaSDI.

simultaneously, enabling identification of simple latent-space dynamics for improved accuracy and efficiency of data-driven computing. To maximize and accelerate the exploration of the parameter space, we introduce an adaptive greedy sampling algorithm integrated with a physics-informed residual-based error indicator and random-subset evaluation to search for the optimal training samples on-the-fly. Moreover, an efficient *k*-nearest neighbor convex interpolation scheme is employed for model evaluation to exploit local latent-space dynamics captured by the local DI models.

To demonstrate the effectiveness of the proposed gLaSDI framework, it has been applied to model various nonlinear dynamical problems, including 1D Burgers' equations, 2D Burgers' equations, nonlinear heat conduction, and time-dependent radial advection. It is observed that greedy sampling with a small *k* for model evaluation results in a more conservative gLaSDI model at the cost of training efficiency, and that the model testing with a large *k* enhances generalization performance of gLaSDI. Compared with LaSDI that has predefined uniformly distributed training parameters, gLaSDI with adaptive and sparse sampling can intelligently identify the optimal training parameter points to achieve higher accuracy with less number of training points than LaSDI. Owning to interactive and simultaneous training of the autoencoder and DI models, gLaSDI is able to capture simpler and smoother latent-space dynamics than LaSDI that has sequential and decoupled training of the autoencoder and DI models. In the radial advection problem, it is also shown that gLaSDI remains highly accurate as the parameter space increases, whereas LaSDI's performances could be deteriorated tremendously. In the numerical examples, compared with the high-fidelity models, gLaSDI achieves 66 to 4,417$\times$ speed-up, with 1 to 5% maximum relative errors in the prescribed parameter space, which reveals the promising potential of applying gLaSDI to large-scale physical simulations.

The proposed gLaSDI framework is general and not restricted by the use of autoencoders and DI models. Depending on applications, various linear or nonlinear data compression techniques other than autoencoders could be employed. Further, latent-space dynamics identification could be performed by other system identification techniques or operator learning algorithms.

The autoencoder architecture can be optimized to maximize generalization performance by integrating automatic neural architecture search into the proposed framework. The parameterization in this study only considers the parameters from the initial conditions of the problems. The proposed framework can be easily extended to account for other parameterization types, such as material properties, which will be useful for inverse problems.

8.6 Acknowledgement

This chapter, in part, has been submitted for publication of the material as it appears in: "Xiaolong He, Youngsoo Choi, William D. Fries, Jon Belof, and Jiun-Shyan Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 2022". The dissertation author is the primary investigator and author of this paper.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In this dissertation, we aim to enhance predictivity and efficiency of physical simulations by developing *thermodynamically consistent* data-driven computing, materials modeling, and reduced-order modeling methods based on emerging machine learning techniques for manifold learning, dimensionality reduction, sequence learning, and system identification.

For *reversible* mechanical systems, we first developed a new data-driven material solver built upon the local convexity-preserving reconstruction scheme [38] to capture anisotropic material behaviors and enable data-driven modeling of anisotropic nonlinear elastic solids. The proposed data-driven approach assumes that the material data of anisotropic materials with a specific anisotropic orientation in a reference frame is accessible. The information of anisotropic orientations, e.g., the rotation angles between local fiber frames and the reference frame of the material data are utilized to construct an offline material database, which contains rotated material data sets representing anisotropic material properties with various anisotropic orientations. The offline rotated material database can be efficiently constructed and applied to data-driven simulations of anisotropic materials. The performance of the proposed data-driven computing framework is demonstrated by effectively modeling deflection of a multi-layer anisotropic cantilever beam made of materials with different anisotropic orientations and inflation of an anisotropic cylinder where anisotropic orientations are along the circumferential direction of the cylinder.

To overcome the curse of dimensionality and the lack of generalization in classical model-free data-driven computing approaches, we introduced a deep manifold learning approach via autoencoders to learn the underlying material data structure and incorporated it into the data-driven solver to enhance solution accuracy, generalization ability, efficiency, and robustness in data-driven computing. In the proposed auto-embedding data-driven (AEDD) computing approach, autoencoders are trained in an offline stage and thus consume little computational overhead in solution procedures. The trained autoencoders are then applied in the proposed data-driven solver during online computation. The trained encoders and decoders define the explicit transformation between low- and high-dimensional spaces of material data, enabling efficient embedding extension to new data points. A simple Shepard convex interpolation scheme is employed in the proposed data-driven solver to preserve convexity in the local data reconstruction, enhancing the robustness of the data-driven solver. The effectiveness of the proposed AEDD framework is examined by modeling biological tissues using experimental data, which shows stronger generalization capability and robustness than the LCDD approach [47].

For *irreversible* mechanical systems, we developed a thermodynamically consistent machine-learned internal state variable (ISV) approach for data-driven modeling of *irreversible* path-dependent materials, which relies purely on the measurable material states. The proposed TCRNN constitutive models consist of two main components: an RNN that infers ISVs and describes their evolution by following the thermodynamics second law, and a DNN that predicts the Helmholtz free energy given strain, ISVs, and temperature (for non-isothermal processes). Two TCRNN constitutive models are developed, one based on the time rates of ISVs and the other one based on the increments of ISVs. The latter model shows an enhanced efficiency as it utilizes an approximation of time rates of ISVs for the calculation of dissipation rate and avoids time-consuming differentiation of the RNN outputs with respect to all RNN inputs. Model robustness and accuracy is enhanced by introducing *stochasticity* to the training data to account for uncertainties of input conditions in the testing. The effectiveness of the proposed TCRNN constitutive model is demonstrated by modeling undrained soil under cyclic shear loading using experimental data, where only measurable material states (stresses and strains) are available. The generalization capability of the TCRNN constitutive model is demonstrated by the effective prediction of the thermodynamically consistent response of undrained soil under the loading conditions different from the ones used in training, which reveals the promising potential of the proposed method to model complex path-dependent materials behaviors in real applications.

Lastly, we developed a physics-informed greedy latent-space dynamics identification (gLaSDI) framework for non-intrusive accurate and efficient data-driven reduced-order modeling of general high-dimensional nonlinear dynamical systems. The proposed gLaSDI framework is composed of an autoencoder that performs nonlinear compression of high-dimensional data and discovers intrinsic latent representations as well as dynamics identification (DI) models that capture local latent-space dynamics. The autoencoder and DI models are trained interactively and simultaneously, enabling identification of simple latent-space dynamics for improved accuracy and efficiency of data-driven computing. To maximize and accelerate the exploration of the parameter space, we introduce an adaptive greedy sampling algorithm integrated with a physics-informed residual-based error indicator and random-subset evaluation to search for the optimal training samples on the fly. Moreover, an efficient k-nearest neighbor convex interpolation scheme is employed for model evaluation to exploit local latent-space dynamics captured by the local DI models. In the numerical examples, compared with the high-fidelity models, gLaSDI achieves 66 to 4,417$\times$ speed-up, with 1 to 5% maximum relative errors in the prescribed parameter space, which reveals the promising potential of applying gLaSDI to large-scale physical simulations.

9.2 Future Work

The recommendations for future work are summarized as follows.

- Development of more sophisticated metrics for anisotropic distance function and reconstruction schemes in order to achieve high accuracy in reconstructing anisotropic material properties from given material data sets that have a large anisotropic distance.

- Application of the proposed data-driven anisotropic modeling framework with real-world data on three-dimensional material systems with aniostropic material behaviors, e.g., musculoskeletal systems consisting of muscle fibers with varying anisotropic orientations.
- Investigation of automatically determining the optimal number of nearest neighbors and the weight matrix in the data-driven local solver to achieve accurate and robust data-driven solutions.
- Integration of uncertainty quantification into the proposed TCRNN model to inform reliability of model predictions.
- Investigation of more efficient training strategies to counteract errors caused by *teacher forcing* training of the proposed TCRNN constitutive model.
- Application of the TCRNN constitutive model to accelerate large-scale multi-scale simulations with complex microstructures and path-dependent material systems
- Optimizing the autoencoder architecture in the proposed gLaSDI framework to maximize generalization performance by applying automatic neural architecture search.
- Extension of the proposed gLaSDI framework to account for other types of parameterization, e.g., material properties and boundary conditions.

Bibliography

[1] Tariq M Khan and Antonio Robles-Kelly. Machine learning: Quantum vs classical. *IEEE Access*, 8:219275–219294, 2020.

[2] Trenton Kirchdoerfer and Michael Ortiz. Data-driven computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 304:81–101, 2016.

[3] Qizhi He and Jiun-Shyan Chen. A physics-constrained data-driven approach based on locally convex reconstruction for noisy database. *Computer Methods in Applied Mechanics and Engineering*, 363:112791, 2020.

[4] Xiaolong He and Jiun-Shyan Chen. Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, page 115348, 2022.

[5] Xiaolong He, Qizhi He, Jiun-Shyan Chen, Usha Sinha, and Shantanu Sinha. Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids. *Data-Centric Engineering*, 1, 2020.

[6] Xiaolong He, Qizhi He, and Jiun-Shyan Chen. Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Computer Methods in Applied Mechanics and Engineering*, 385:114034, 2021.

[7] Xiaolong He, Youngsoo Choi, William D. Fries, Jon Belof, and Jiun-Shyan Chen. glasdi: Parametric physics-informed greedy latent space dynamics identification. *Journal of Computational Physics*, 2022.

[8] Karan Taneja, Xiaolong He, Qizhi He, Zhao Xinlun, Lin Yun-An, Loh Kenneth, and Jiun-Shyan Chen. A feature-encoded physics-informed parameter identification neural network for musculo-skeletal systems. *Journal of Biomechanical Engineering*, 2022.

[9] William Fries, Xiaolong He, and Youngsoo Choi. Lasdi: Parametric latent space dynamics identification. *Computer Methods in Applied Mechanics and Engineering*, 2022.

[10] Yu-Yen Chen, M Ross Kunz, Xiaolong He, and Rebecca Fushimi. Recent progress toward

catalyst properties, performance, and prediction with data-driven methods. *Current Opinion in Chemical Engineering*, 37:100843, 2022.

[11] M Ross Kunz, Adam Yonge, Xiaolong He, Rakesh Batchu, Zongtang Fang, Yixiao Wang, Gregory S Yablonsky, Andrew J Medford, and Rebecca R Fushimi. Internal calibration of transient kinetic data via machine learning. *Catalysis Today*, 2022.

[12] Xiaolong He, Karan Taneja, Jiun-Shyan Chen, Chung-Hao Lee, John Hodgson, Vadim Malis, Usha Sinha, and Shantanu Sinha. Multiscale modeling of passive material influences on deformation and force output of skeletal muscles. *International Journal for Numerical Methods in Biomedical Engineering*, page e3571, 2022.

[13] Benjamin Reedlunn, Georgios Moutsanidis, Jonghyuk Baek, Tsung-Hui Huang, Jacob Koester, Xiaolong He, Haoyan Wei, Karan Taneja, Yuri Bazilevs, and Jiun-Shyan Chen. Initial simulations of empty room collapse and reconsolidation at the waste isolation pilot plant. In *54th US Rock Mechanics/Geomechanics Symposium*. OnePetro, 2020.

[14] Yantao Zhang, Jiun-Shyan Chen, Qizhi He, Xiaolong He, Ramya R Basava, John Hodgson, Usha Sinha, and Shantanu Sinha. Microstructural analysis of skeletal muscle force generation during aging. *International journal for numerical methods in biomedical engineering*, 36(1):e3295, 2020.

[15] Lingling Hu, Danyang Cai, Gangpeng Wu, Xiaolong He, and Tongxi Yu. Influence of internal pressure on the out-of-plane dynamic behavior of circular-celled honeycombs. *International Journal of Impact Engineering*, 104:64–74, 2017.

[16] Changyong Jiang, Xiaolong He, and Lixi Huang. Equivalent acoustic parameters in a periodical waveguide structure. In *Fluid-Structure-Sound Interactions and Control*, pages 83–88. Springer, 2016.

[17] LingLling Hu, Xiaolong He, Gangpeng Wu, and Tongxi Yu. Dynamic crushing of the circular-celled honeycombs under out-of-plane impact. *International Journal of Impact Engineering*, 75:150–161, 2015.

[18] Peter A Huijing. Muscle as a collagen fiber reinforced composite: a review of force transmission in muscle and whole limb. *Journal of biomechanics*, 32(4):329–345, 1999.

[19] Allison R Gillies and Richard L Lieber. Structure and function of the skeletal muscle extracellular matrix. *Muscle & nerve*, 44(3):318–331, 2011.

[20] Michael Takaza, Kevin M Moerman, Juliette Gindre, Garry Lyons, and Ciaran K Simms. The anisotropic mechanical behaviour of passive skeletal muscle tissue subjected to large tensile strain. *Journal of the mechanical behavior of biomedical materials*, 17:209–220, 2013.

[21] Bernard D Coleman and Morton E Gurtin. Thermodynamics with internal state variables. *The journal of chemical physics*, 47(2):597–613, 1967.

[22] Mark F Horstemeyer and Douglas J Bammann. Historical review of internal state variable theory for inelasticity. *International Journal of Plasticity*, 26(9):1310–1334, 2010.

[23] J Kratochvil and OW Dillon Jr. Thermodynamics of elastic-plastic materials as a theory with internal state variables. *Journal of Applied Physics*, 40(8):3207–3218, 1969.

[24] JC Simo and Ch Miehe. Associative coupled thermoplasticity at finite strains: Formulation, numerical analysis and implementation. *Computer Methods in Applied Mechanics and Engineering*, 98(1):41–104, 1992.

[25] Juan C Simo. Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory. *Computer Methods in Applied Mechanics and Engineering*, 99(1):61–112, 1992.

[26] Piotr Perzyna. Internal state variable description of dynamic fracture of ductile solids. *International Journal of Solids and Structures*, 22(7):797–818, 1986.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[28] Rubén Ibanez, Emmanuelle Abisset-Chavanne, Jose Vicente Aguado, David Gonzalez, Elias Cueto, and Francisco Chinesta. A manifold learning approach to data-driven computational elasticity and inelasticity. *Archives of Computational Methods in Engineering*, 25(1):47–57, 2018.

[29] Shun Wang, Eric de Sturler, and Glaucio H Paulino. Large-scale topology optimization using preconditioned krylov subspace methods with recycling. *International journal for numerical methods in engineering*, 69(12):2441–2468, 2007.

[30] Daniel A White, Youngsoo Choi, and Jun Kudo. A dual mesh method with adaptivity for stress-constrained topology optimization. *Structural and Multidisciplinary Optimization*, 61(2):749–762, 2020.

[31] Youngsoo Choi, Charbel Farhat, Walter Murray, and Michael Saunders. A practical factorization of a schur complement for pde-constrained distributed optimal control. *Journal of Scientific Computing*, 65(2):576–597, 2015.

[32] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*, volume 12. Siam, 2013.

[33] George Biros, Omar Ghattas, Matthias Heinkenschloss, David Keyes, Bani Mallick, Luis Tenorio, Bart van Bloemen Waanders, Karen Willcox, Youssef Marzouk, and Lorenz

Biegler. *Large-scale inverse problems and quantification of uncertainty*. John Wiley & Sons, 2011.

[34] David Galbally, Krzysztof Fidkowski, Karen Willcox, and Omar Ghattas. Non-linear model reduction for uncertainty quantification in large-scale inverse problems. *International journal for numerical methods in engineering*, 81(12):1581–1608, 2010.

[35] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[36] Anthony T Patera, Gianluigi Rozza, et al. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations, 2007.

[37] Michael G Safonov and RY1000665 Chiang. A schur method for balanced-truncation model reduction. *IEEE Transactions on Automatic Control*, 34(7):729–733, 1989.

[38] Qizhi He and Jiun-Shyan Chen. A physics-constrained data-driven approach based on locally convex reconstruction for noisy database. *Computer Methods in Applied Mechanics and Engineering*, 363:112791, 2020.

[39] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[40] Gal Mishne, Uri Shaham, Alexander Cloninger, and Israel Cohen. Diffusion nets. *Applied and Computational Harmonic Analysis*, 47(2):259–285, 2019.

[41] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[42] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[43] Palash Goyal, Sumit Pandey, and Karan Jain. Deep learning for natural language processing. *New York: Apress*, 2018.

[44] David DeMers and Garrison W Cottrell. Non-linear dimensionality reduction. In *Advances in neural information processing systems*, pages 580–587, 1993.

[45] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[46] John A Lee and Michel Verleysen. *Nonlinear Dimensionality Reduction*. Springer Science & Business Media, 2007.

[47] Qizhi He, Devin W Laurence, Chung-Hao Lee, and Jiun-Shyan Chen. Manifold learning based data-driven modeling for soft biological tissues. *Journal of Biomechanics*, 117:110124, 2020.

[48] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[49] QiZhi He and Alexandre M Tartakovsky. Physics-informed neural network method for forward and backward advection-dispersion equations. *Water Resources Research*, 57(7):e2020WR029479, 2021.

[50] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[51] Teeratorn Kadeethum, Daniel O'Malley, Jan Niklas Fuhg, Youngsoo Choi, Jonghyun Lee, Hari S Viswanathan, and Nikolaos Bouklas. A framework for data-driven solution and parameter estimation of pdes using conditional generative adversarial networks. *Nature Computational Science*, 1(12):819–829, 2021.

[52] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[53] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33:17429–17442, 2020.

[54] Alexandre M Tartakovsky, C Ortiz Marrero, Paris Perdikaris, Guzel D Tartakovsky, and David Barajas-Solano. Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5):e2019WR026731, 2020.

[55] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

[56] QiZhi He, Panos Stinis, and Alexandre M Tartakovsky. Physics-constrained deep neural network method for estimating parameters in a redox flow battery. *Journal of Power Sources*, 528:231147, 2022.

[57] Xuping Xie, Guannan Zhang, and Clayton G Webster. Non-intrusive inference reduced order model for fluids using deep multistep neural network. *Mathematics*, 7(8):757, 2019.

[58] Zhe Bai and Liqian Peng. Non-intrusive nonlinear model reduction via machine learning approximations to low-dimensional operators. *Advanced Modeling and Simulation in Engineering Sciences*, 8(1):1–24, 2021.

[59] Shigeki Kaneko, Haoyan Wei, Qizhi He, Jiun-Shyan Chen, and Shinobu Yoshimura. A hyper-reduction computational method for accelerated modeling of thermal cycling-induced plastic deformations. *Journal of the Mechanics and Physics of Solids*, 151:104385, 2021.

[60] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

[61] Teeratorn Kadeethum, Francesco Ballarin, Youngsoo Choi, Daniel O'Malley, Hongkyu Yoon, and Nikolaos Bouklas. Non-intrusive reduced order modeling of natural convection in porous media using convolutional autoencoders: comparison with linear subspace techniques. *Advances in Water Resources*, page 104098, 2022.

[62] Miguel A Bessa, R Bostanabad, Zeliang Liu, A Hu, Daniel W Apley, C Brinson, Wei Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering*, 320:633–667, 2017.

[63] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[64] Yujie Zhang and Wenjing Ye. Deep learning–based inverse method for layout design. *Structural and Multidisciplinary Optimization*, 60(2):527–536, 2019.

[65] Xin Lei, Chang Liu, Zongliang Du, Weisheng Zhang, and Xu Guo. Machine learning-driven real-time topology optimization under moving morphable component-based framework. *Journal of Applied Mechanics*, 86(1), 2019.

[66] Jonghyuk Baek, Jiun-Shyan Chen, and Kristen Susuki. A neural network-enhanced reproducing kernel particle method for modeling strain localization. *International Journal for Numerical Methods in Engineering*, 2022.

[67] Frederic E Bock, Roland C Aydin, Christian Johannes Cyron, Norbert Huber, Surya R Kalidindi, and Benjamin Klusemann. A review of the application of machine learning and data mining approaches in continuum materials mechanics. *Frontiers in Materials*, 6:110, 2019.

[68] J Ghaboussi, JH Garrett Jr, and Xiping Wu. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.

[69] Yuelin Shen, K Chandrashekhara, WF Breig, and LR Oliver. Finite element analysis of v-ribbed belts using neural network based hyperelastic material model. *International Journal of Non-Linear Mechanics*, 40(6):875–890, 2005.

[70] Tomonari Furukawa and Genki Yagawa. Implicit constitutive modelling for viscoplasticity using neural networks. *International Journal for Numerical Methods in Engineering*, 43(2):195–219, 1998.

[71] M Lefik, DP Boso, and BA Schrefler. Artificial neural networks in numerical modelling of composites. *Computer Methods in Applied Mechanics and Engineering*, 198(21-26):1785–1804, 2009.

[72] Marek Lefik and Bernhard A Schrefler. Artificial neural network as an incremental non-linear constitutive model for a finite element code. *Computer methods in applied mechanics and engineering*, 192(28-30):3265–3283, 2003.

[73] YMA Hashash, S Jung, and J Ghaboussi. Numerical implementation of a neural network based material model in finite element analysis. *International Journal for numerical methods in engineering*, 59(7):989–1005, 2004.

[74] Sungmoon Jung and Jamshid Ghaboussi. Neural network constitutive model for rate-dependent materials. *Computers & Structures*, 84(15-16):955–963, 2006.

[75] Marcus Stoffel, Franz Bamer, and Bernd Markert. Neural network based constitutive modeling of nonlinear viscoplastic structural response. *Mechanics Research Communications*, 95:85–88, 2019.

[76] Annan Zhang and Dirk Mohr. Using neural networks to represent von mises plasticity with isotropic hardening. *International Journal of Plasticity*, 132:102732, 2020.

[77] Kailai Xu, Daniel Z Huang, and Eric Darve. Learning constitutive relations using symmetric positive definite neural networks. *Journal of Computational Physics*, 428:110072, 2021.

[78] Julia Ling, Reese Jones, and Jeremy Templeton. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016.

[79] Nikolaos N Vlassis and WaiChing Sun. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, 377:113695, 2021.

[80] Filippo Masi, Ioannis Stefanou, Paolo Vannucci, and Victor Maffi-Berthier. Thermodynamics-based artificial neural networks for constitutive modeling. *Journal of the Mechanics and Physics of Solids*, 147:104277, 2021.

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

[82] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[83] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[84] Yousef Heider, Kun Wang, and WaiChing Sun. So (3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials. *Computer Methods in Applied Mechanics and Engineering*, 363:112875, 2020.

[85] M Mozaffar, R Bostanabad, W Chen, K Ehmann, Jian Cao, and MA Bessa. Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences*, 116(52):26414–26420, 2019.

[86] Guang Chen. Recurrent neural networks (rnns) learn the constitutive law of viscoelasticity. *Computational Mechanics*, 67(3):1009–1019, 2021.

[87] Maysam B Gorji, Mojtaba Mozaffar, Julian N Heidenreich, Jian Cao, and Dirk Mohr. On the potential of recurrent neural networks for modeling path dependent plasticity. *Journal of the Mechanics and Physics of Solids*, 143:103972, 2020.

[88] Kun Wang and WaiChing Sun. A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning. *Computer Methods in Applied Mechanics and Engineering*, 334:337–380, 2018.

[89] F Ghavamian and A Simone. Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. *Computer Methods in Applied Mechanics and Engineering*, 357:112594, 2019.

[90] Ling Wu, Nanda Gopala Kilingar, Ludovic Noels, et al. A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths. *Computer Methods in Applied Mechanics and Engineering*, 369:113234, 2020.

[91] Hernan J Logarzo, German Capuano, and Julian J Rimoli. Smart constitutive laws: Inelas-

tic homogenization through machine learning. *Computer methods in applied mechanics and engineering*, 373:113482, 2021.

[92] Ling Wu and Ludovic Noels. Recurrent neural networks (rnns) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step. *Computer Methods in Applied Mechanics and Engineering*, 390:114476, 2022.

[93] Colin Bonatti and Dirk Mohr. On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids. *Journal of the Mechanics and Physics of Solids*, 158:104697, 2022.

[94] S. Conti, S. Müller, and M. Ortiz. Data-Driven Problems in Elasticity. *Archive for Rational Mechanics and Analysis*, 229(1):79–123, 2018.

[95] Trenton Kirchdoerfer and Michael Ortiz. Data-driven computing in dynamics. *International Journal for Numerical Methods in Engineering*, 113(11):1697–1710, 2018.

[96] Lu Trong Khiem Nguyen and Marc-André Keip. A data-driven approach to nonlinear elasticity. *Computers & Structures*, 194:97–115, 2018.

[97] Robert Eggersmann, Trenton Kirchdoerfer, Stefanie Reese, Laurent Stainier, and Michael Ortiz. Model-free data-driven inelasticity. *Computer Methods in Applied Mechanics and Engineering*, 350:81–99, 2019.

[98] Ruben Ibañez, Domenico Borzacchiello, Jose Vicente Aguado, Emmanuelle Abisset-Chavanne, Elias Cueto, Pierre Ladeveze, and Francisco Chinesta. Data-driven non-linear elasticity: constitutive manifold construction and problem discretization, 2017.

[99] Adrien Leygue, Michel Coret, Julien Réthoré, Laurent Stainier, and Erwan Verron. Data-based derivation of material response. *Computer Methods in Applied Mechanics and Engineering*, 331:184–196, 2018.

[100] Jacobo Ayensa-Jiménez, Mohamed H. Doweidar, Jose A. Sanz-Herrera, and Manuel Doblaré. An unsupervised data completion method for physically-based data-driven models. *Computer Methods in Applied Mechanics and Engineering*, 344:120–143, 2019.

[101] Laurent Stainier, Adrien Leygue, and Michael Ortiz. Model-free data-driven methods in mechanics: material data identification and solvers. *Computational Mechanics*, 64(2):381–393, 2019.

[102] Lu Trong Khiem Nguyen, Matthias Rambausek, and Marc André Keip. Variational framework for distance-minimizing method in data-driven computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 365:112898, 2020.

[103] Trenton Kirchdoerfer and Michael Ortiz. Data driven computing with noisy material data sets. *Computer Methods in Applied Mechanics and Engineering*, 326:622–641, 2017.

[104] Jacobo Ayensa-Jiménez, Mohamed H Doweidar, Jose A Sanz-Herrera, and Manuel Doblaré. A new reliability-based data-driven approach for noisy experimental data with physical constraints. *Computer Methods in Applied Mechanics and Engineering*, 328:752–774, 2018.

[105] Robert Eggersmann, Laurent Stainier, Michael Ortiz, and Stefanie Reese. Model-free data-driven compuational mechanics enhanced by tensor voting. *arXiv preprint arXiv:2004.02503*, 2020.

[106] Christoph Settgast, Geralf Hütter, Meinhard Kuna, and Martin Abendroth. A hybrid approach to simulate the homogenized irreversible elastic–plastic deformations and damage of foams by neural networks. *International Journal of Plasticity*, 126(May):102624, 2019.

[107] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020.

[108] Mengnan Liu, Shuiliang Fang, Huiyue Dong, and Cunzhi Xu. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361, 2021.

[109] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. Efficient nonlinear manifold reduced order model. *arXiv preprint arXiv:2011.07727*, 2020.

[110] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[111] Chi Hoang, Youngsoo Choi, and Kevin Carlberg. Domain-decomposition least-squares petrov–galerkin (dd-lspg) nonlinear model reduction. *Computer Methods in Applied Mechanics and Engineering*, 384:113997, 2021.

[112] Dylan Matthew Copeland, Siu Wun Cheung, Kevin Huynh, and Youngsoo Choi. Reduced order models for lagrangian hydrodynamics. *Computer Methods in Applied Mechanics and Engineering*, 388:114259, 2022.

[113] Siu Wun Cheung, Youngsoo Choi, Dylan Matthew Copeland, and Kevin Huynh. Local lagrangian reduced-order modeling for rayleigh-taylor instability by solution manifold decomposition. *arXiv preprint arXiv:2201.07335*, 2022.

[114] Jessica Lauzon, Siu Wun Cheung, Yeonjong Shin, Youngsoo Choi, Matthew Copeland,

and Kevin Huynh. S-OPT: A points selection algorithm for hyper-reduction in reduced order models. *arXiv preprint arXiv:2203.16494*, 2022.

[115] Felix Fritzen, Bernard Haasdonk, David Ryckelynck, and Sebastian Schöps. An algorithmic comparison of the hyper-reduction and the discrete empirical interpolation method for a nonlinear thermal problem. *Mathematical and computational applications*, 23(1):8, 2018.

[116] Youngsoo Choi, Deshawn Coombs, and Robert Anderson. Sns: A solution-based nonlinear subspace method for time-dependent model order reduction. *SIAM Journal on Scientific Computing*, 42(2):A1116–A1146, 2020.

[117] Youngsoo Choi and Kevin Carlberg. Space–time least-squares petrov–galerkin projection for nonlinear model reduction. *SIAM Journal on Scientific Computing*, 41(1):A26–A58, 2019.

[118] Kevin Carlberg, Youngsoo Choi, and Syuzanna Sargsyan. Conservative model reduction for finite-volume models. *Journal of Computational Physics*, 371:280–314, 2018.

[119] Benjamin McLaughlin, Janet Peterson, and Ming Ye. Stabilized reduced order models for the advection–diffusion–reaction equation using operator splitting. *Computers & Mathematics with Applications*, 71(11):2407–2420, 2016.

[120] Youngkyu Kim, Karen Wang, and Youngsoo Choi. Efficient space–time reduced order model for linear dynamical systems in python using less than 120 lines of code. *Mathematics*, 9(14):1690, 2021.

[121] Giovanni Stabile and Gianluigi Rozza. Finite volume pod-galerkin stabilised reduced order methods for the parametrised incompressible navier–stokes equations. *Computers & Fluids*, 173:273–284, 2018.

[122] Traian Iliescu and Zhu Wang. Variational multiscale proper orthogonal decomposition: Navier-stokes equations. *Numerical Methods for Partial Differential Equations*, 30(2):641–663, 2014.

[123] Alexander C Hughes and Andrew G Buchan. A discontinuous and adaptive reduced order model for the angular discretization of the boltzmann transport equation. *International Journal for Numerical Methods in Engineering*, 121(24):5647–5666, 2020.

[124] Youngsoo Choi, Peter Brown, William Arrighi, Robert Anderson, and Kevin Huynh. Space–time reduced order model for large-scale linear dynamical systems with application to boltzmann transport problems. *Journal of Computational Physics*, 424:109845, 2021.

[125] Jiun-Shyan Chen, Camille Marodon, and Hsin-Yun Hu. Model order reduction for

meshfree solution of poisson singularity problems. *International Journal for Numerical Methods in Engineering*, 102(5):1211–1237, 2015.

[126] Qizhi He, Jiun-Shyan Chen, and Camille Marodon. A decomposed subspace reduction for fracture mechanics based on the meshfree integrated singular basis function method. *Computational Mechanics*, 63(3):593–614, 2019.

[127] Chung-Hao Lee and Jiun-Shyan Chen. Proper orthogonal decomposition-based model order reduction via radial basis functions for molecular dynamics systems. *International journal for numerical methods in engineering*, 96(10):599–627, 2013.

[128] Chung-Hao Lee and Jiun-Shyan Chen. RBF-POD reduced-order modeling of DNA molecules under stretching and bending. *Interaction and Multiscale Mechanics*, 6(4):395–409, 2013.

[129] Christian Gogu. Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction. *International Journal for Numerical Methods in Engineering*, 101(4):281–304, 2015.

[130] Youngsoo Choi, Geoffrey Oxberry, Daniel White, and Trenton Kirchdoerfer. Accelerating design optimization using reduced order models. *arXiv preprint arXiv:1909.11320*, 2019.

[131] Sean McBane and Youngsoo Choi. Component-wise reduced order model lattice-type structure design. *Computer Methods in Applied Mechanics and Engineering*, 381:113813, 2021.

[132] Youngsoo Choi, Gabriele Boncoraglio, Spenser Anderson, David Amsallem, and Charbel Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *Journal of Computational Physics*, 423:109787, 2020.

[133] Hannah Lu and Daniel M Tartakovsky. Dynamic mode decomposition for construction of reduced-order models of hyperbolic problems with shocks. *Journal of Machine Learning for Modeling and Computing*, 2(1), 2021.

[134] Hannah Lu and Daniel M Tartakovsky. Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena. *Journal of Computational Physics*, 407:109229, 2020.

[135] Rambod Mojgani and Maciej Balajewicz. Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows. *arXiv preprint arXiv:1701.04343*, 2017.

[136] Julius Reiss, Philipp Schulze, Jörn Sesterhenn, and Volker Mehrmann. The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena.

SIAM Journal on Scientific Computing, 40(3):A1322–A1344, 2018.

[137] Marzieh Alireza Mirhoseini and Matthew J Zahr. Model reduction of convection-dominated partial differential equations via optimization-based implicit feature tracking. *arXiv preprint arXiv:2109.14694*, 2021.

[138] Davide Papapicco, Nicola Demo, Michele Girfoglio, Giovanni Stabile, and Gianluigi Rozza. The neural network shifted-proper orthogonal decomposition: A machine learning approach for non-linear reduction of hyperbolic equations. *Computer Methods in Applied Mechanics and Engineering*, 392:114687, 2022.

[139] Benjamin Peherstorfer. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *SIAM Journal on Scientific Computing*, 42(5):A2803–A2836, 2020.

[140] Angelo Iollo and Damiano Lombardi. Advection modes by optimal mass transfer. *Physical Review E*, 89(2):022923, 2014.

[141] Zhiguang Qian, Carolyn Conner Seepersad, V Roshan Joseph, Janet K Allen, and CF Jeff Wu. Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, 128(4):668–677, 2006.

[142] Gustavo Tapia, Saad Khairallah, Manyalibo Matthews, Wayne E King, and Alaa Elwany. Gaussian process-based surrogate modeling framework for process planning in laser powder-bed fusion additive manufacturing of 316l stainless steel. *The International Journal of Advanced Manufacturing Technology*, 94(9):3591–3603, 2018.

[143] B Daniel Marjavaara, T Staffan Lundström, Tushar Goel, Yolanda Mack, and Wei Shyy. Hydraulic turbine diffuser shape optimization by multiple surrogate model approximations of pareto fronts. *Journal of Fluids Engineering*, 129(9):1228–1240, 2007.

[144] Fuxin Huang, Lijue Wang, and Chi Yang. Hull form optimization for reduced drag and improved seakeeping using a surrogate-based method. In *The Twenty-fifth International Ocean and Polar Engineering Conference*. OnePetro, 2015.

[145] Zhong-Hua Han, Stefan Görtz, and Ralf Zimmermann. Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function. *Aerospace Science and technology*, 25(1):177–189, 2013.

[146] Zhong-Hua Han and Stefan Görtz. Hierarchical kriging model for variable-fidelity surrogate modeling. *AIAA journal*, 50(9):1885–1896, 2012.

[147] J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[148] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Calogan: Simulating 3d high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Physical Review D*, 97(1):014021, 2018.

[149] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869, 2017.

[150] Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018.

[151] T Kadeethum, D O'Malley, Y Choi, HS Viswanathan, N Bouklas, and H Yoon. Continuous conditional generative adversarial networks for data-driven solutions of poroelasticity with heterogeneous material properties. *arXiv preprint arXiv:2111.14984*, 2021.

[152] Teeratorn Kadeethum, Francesco Ballarin, Daniel O'Malley, Youngsoo Choi, Nikolaos Bouklas, and Hongkyu Yoon. Reduced order modeling for flow and transport problems with barlow twins self-supervised learning. *arXiv preprint arXiv:2202.05460*, 2022.

[153] Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, and Karen Willcox. Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*, 179:704–717, 2019.

[154] Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pages 59–70. Wiley Online Library, 2019.

[155] Chi Hoang, Kenny Chowdhary, Kookjin Lee, and Jaideep Ray. Projection-based model reduction of dynamical systems using space–time subspace and machine learning. *Computer Methods in Applied Mechanics and Engineering*, 389:114341, 2022.

[156] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[157] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[158] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[159] Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica*

D: Nonlinear Phenomena, 406:132401, 2020.

[160] Peter Benner, Pawan Goyal, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Computer Methods in Applied Mechanics and Engineering*, 372:113433, 2020.

[161] M Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, 2020.

[162] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

[163] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pages 1945–1954. PMLR, 2017.

[164] Li Li, Minjie Fan, Rishabh Singh, and Patrick Riley. Neural-guided symbolic regression with asymptotic constraints. *arXiv preprint arXiv:1901.07714*, 2019.

[165] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[166] Rudy Geelen, Stephen Wright, and Karen Willcox. Operator inference for non-intrusive model reduction with nonlinear manifolds. *arXiv preprint arXiv:2205.02304*, 2022.

[167] Mengwu Guo, Shane McQuarrie, and Karen Willcox. Bayesian operator inference for data-driven reduced order modeling. *arXiv preprint arXiv:2204.10829*, 2022.

[168] Rudy Geelen and Karen Willcox. Localized non-intrusive reduced-order modeling in the operator inference framework. *Philosophical Transactions of the Royal Society A, to appear*, 2022.

[169] Shane A McQuarrie, Parisa Khodabakhshi, and Karen E Willcox. Non-intrusive reduced-order models for parametric partial differential equations via data-driven operator inference. *arXiv preprint arXiv:2110.07653*, 2021.

[170] Renee Swischuk, Boris Kramer, Cheng Huang, and Karen Willcox. Learning physics-based reduced-order models for a single-injector combustion process. *AIAA Journal*, 58(6):2658–2672, 2020.

[171] Parikshit Jain, Shane McQuarrie, and Boris Kramer. Performance comparison of data-driven reduced models for a single-injector combustion process. In *AIAA Propulsion and*

Energy 2021 Forum, page 3633, 2021.

[172] Shane A McQuarrie, Cheng Huang, and Karen E Willcox. Data-driven reduced-order models via regularised operator inference for a single-injector combustion process. *Journal of the Royal Society of New Zealand*, 51(2):194–211, 2021.

[173] Benjamin Peherstorfer. Sampling low-dimensional markovian dynamics for preasymptotically recovering reduced models from data with operator inference. *SIAM Journal on Scientific Computing*, 42(5):A3489–A3515, 2020.

[174] Parisa Khodabakhshi and Karen E Willcox. Non-intrusive data-driven model reduction for differential–algebraic equations derived from lifting transformations. *Computer Methods in Applied Mechanics and Engineering*, 389:114296, 2022.

[175] Süleyman Yıldız, Pawan Goyal, Peter Benner, and Bülent Karasözen. Learning reduced-order dynamics for parametrized shallow water equations from data. *International Journal for Numerical Methods in Fluids*, 93(8):2803–2821, 2021.

[176] Opal Issan and Boris Kramer. Predicting solar wind streams from the inner-heliosphere to earth via shifted operator inference. *arXiv preprint arXiv:2203.13372*, 2022.

[177] Ellad B Tadmor, Ronald E Miller, and Ryan S Elliott. *Continuum mechanics and thermodynamics: from fundamental concepts to governing equations*. Cambridge University Press, 2012.

[178] Miroslav Silhavy. *The mechanics and thermodynamics of continuous media*. Springer Science & Business Media, 2013.

[179] Juan C Simo and Thomas JR Hughes. *Computational inelasticity*, volume 7. Springer Science & Business Media, 2006.

[180] Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil Elkhodary. *Nonlinear finite elements for continua and structures*. John wiley & sons, 2013.

[181] Theodore Sussman and Klaus Jürgen Bathe. A model of incompressible isotropic hyperelastic material behavior using spline interpolations of tension-compression test data. *Communications in Numerical Methods in Engineering*, 25(1):53–63, 2009.

[182] Frédéric Feyel. Multiscale FE2 elastoviscoplastic analysis of composite structures. *Computational Materials Science*, 16(1-4):344–354, 1999.

[183] Frédéric Feyel. A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua. *Computer Methods in Applied Mechanics and Engineering*, 192(28-30):3233–3244, 2003.

[184] Marcos Latorre and Francisco Javier Montáns. What-You-Prescribe-Is-What-You-Get orthotropic hyperelasticity. *Computational Mechanics*, 53(6):1279–1298, 2014.

[185] Carlos A. Felippa. A survey of parametrized variational principles and applications to computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 113(1-2):109–139, 1994.

[186] Wing Kam Liu, Sukky Jun, and Yi Fei Zhang. Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106, 1995.

[187] Jiun-Shyan Chen, Chunhui Pan, Cheng-Tang Wu, and Wing Kam Liu. Reproducing kernel particle methods for large deformation analysis of non-linear structures. *Computer methods in applied mechanics and engineering*, 139(1-4):195–227, 1996.

[188] Jiun-Shyan Chen, Sangpil Yoon, and Cheng-Tang Wu. Non-linear version of stabilized conforming nodal integration for galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering*, 53(12):2587–2615, 2002.

[189] Ruben Ibañez, Domenico Borzacchiello, Jose Vicente Aguado, Emmanuelle Abisset-Chavanne, Elías Cueto, Pierre Ladevèze, and Francisco Chinesta. Data-driven non-linear elasticity: constitutive manifold construction and problem discretization. *Computational Mechanics*, 60(5):813–826, 2017.

[190] Laurent Stainier, Adrien Leygue, and Michael Ortiz. Model-free data-driven methods in mechanics: material data identification and solvers. *Computational Mechanics*, 64(2):381–393, 2019.

[191] Rui Xu, Jie Yang, Wei Yan, Qun Huang, Gaetano Giunta, Salim Belouettar, Hamid Zahrouni, Tarak Ben Zineb, and Heng Hu. Data-driven multiscale finite element method: From concurrence to separation. *Computer Methods in Applied Mechanics and Engineering*, 363:112893, 2020.

[192] J Mora-Macías, J Ayensa-Jiménez, E Reina-Romo, MH Doweidar, J Domínguez, M Doblaré, and JA Sanz-Herrera. A multiscale data-driven approach for bone tissue biomechanics. *Computer Methods in Applied Mechanics and Engineering*, 368:113136, 2020.

[193] Ronald K Mitchell, Bradley R Agle, and Donna J Wood. Toward a theory of stakeholder identification and salience: Defining the principle of who and what really counts. *Academy of management review*, 22(4):853–886, 1997.

[194] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[195] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[196] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[197] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[198] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative review. *J Mach Learn Res*, 10(66-71):13, 2009.

[199] I . T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 2002.

[200] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[201] Robert Eggersmann, Laurent Stainier, Michael Ortiz, and Stefanie Reese. Model-free data-driven computational mechanics enhanced by tensor voting. *Computer Methods in Applied Mechanics and Engineering*, 373:113499, 2021.

[202] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[203] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[204] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[205] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(Jan):1–40, 2009.

[206] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968.

[207] Ivo Babuška and Jens M Melenk. The partition of unity method. *International journal for numerical methods in engineering*, 40(4):727–758, 1997.

[208] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press,

2004.

[209] Qizhi He, Zhan Kang, and Yiqiang Wang. A topology optimization method for geometrically nonlinear structures with meshless analysis and independent density field interpolation. *Computational Mechanics*, 54(3):629–644, 2014.

[210] Samuel Jett, Devin Laurence, Robert Kunkel, Anju R Babu, Katherine Kramer, Ryan Baumwart, Rheal Towner, Yi Wu, and Chung-Hao Lee. An investigation of the anisotropic mechanical properties and anatomical structure of porcine atrioventricular heart valves. *Journal of the mechanical behavior of biomedical materials*, 87:155–171, 2018.

[211] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE, 1993.

[212] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[213] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[214] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[215] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[216] Ana Maria Parra Bastidas. *Ottawa F-65 sand characterization*. University of California, Davis, 2016.

[217] Mohamed El Ghoraiby, Hanna Park, and Majid T Manzari. Physical and mechanical properties of ottawa f65 sand. In *Model tests and numerical simulations of liquefaction and lateral spreading*, pages 45–67. Springer, 2020.

[218] W Rong and JS McCartney. Undrained seismic compression of unsaturated sand. *Journal of Geotechnical and Geoenvironmental Engineering*, 147(1):04020145, 2021.

[219] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating*

systems design and implementation (OSDI 16), pages 265–283, 2016.

[220] Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, et al. Mfem: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021.