Universidad Zaragoza
1542

Doctoral Thesis

Structure-Preserving Deep Learning

Author

Quercus Manuel Hernández Laín

Doctoral advisors

Prof. Elías Cueto Prendes
Prof. Alberto Badías Herbera

Ph.D. Programme in **Mechanical Engineering**

September 2023

A mis padres.
A mis abuelos.
A mi tío Jesús.
A mis tíos Carmen y Ambrosio.

Abstract

Simulation technologies have become a useful tool to model many systems in a wide variety of disciplines, ranging from social to natural sciences. In the engineering context, the mathematical description of physical systems is crucial in order to have a deeper understanding of their future behaviour and take the best cost-effective decisions in terms of design optimization. In the last decades, with the irruption of the so-called fourth paradigm of science, a growing interest is detected in the machine learning of these scientific laws. Many of such approaches completely discard centuries of scientific knowledge in favour of pure data-driven models, whose application in real industry remains unclear over traditional mathematically-founded methods.

This thesis aims to develop deep learning methodologies to learn dynamical systems from data. In this work, the scientific knowledge is incorporated into the model with the imposition of the correct thermodynamical structure of the problem. Thus, it is a hybrid approach between data-driven black-box methods and traditional analytical formulations.

The first part of the thesis explores structure-preserving deep learning methods to simulate dynamical systems from data. This is achieved by using several inductive biases, which enforce the metriplectic structure of dynamics by construction and exploit the domain structure using the geometric deep learning principles. These state-of-the-art neural network architectures enable the identification of the dynamics of unknown systems even with highly nonlinear behaviour. Additionally, the resulting integration scheme achieves both fast and accurate predictions with low computational power and memory storage requirements.

The second part describes a method to identify the intrinsic dimensionality of a dynamical system to obtain a reduced order model, which can be integrated in time using a structure-preserving architecture. This method is convenient for high-dimensional data whose complex structure becomes impractical for standard techniques.

The third and final part explores two possible applications of the thesis contributions. A learning procedure is described for open systems in which the interaction of external sources is relevant, based on the port-Hamiltonian formalism and its extension to metriplectic systems. The second application is an augmented reality demo for the interaction of deformable solids in real-time, enabled by the fast predictions of the structure-preserving learnt simulators developed in this thesis.

Resumen

Las tecnologías de simulación se han convertido en una herramienta útil para modelizar muchos sistemas en una amplia variedad de disciplinas, desde las ciencias sociales a las naturales. En el contexto de la ingeniería, la descripción matemática de los sistemas físicos es crucial para comprender mejor su comportamiento futuro y tomar las decisiones más apropiadas en términos de optimización de su diseño. En las últimas décadas, con la irrupción del llamado cuarto paradigma de la ciencia, se ha detectado un creciente interés por el aprendizaje automático de estas leyes científicas. Muchos de estos enfoques descartan por completo siglos de conocimiento científico en favor de modelos basados puramente en datos, cuya aplicación en la industria real sigue sin estar clara frente a los métodos tradicionales con amplio fundamento matemático.

El objetivo de esta tesis es desarrollar metodologías de aprendizaje profundo para aprender sistemas dinámicos a partir de datos. En este trabajo, el conocimiento científico se incorpora al modelo con la imposición de la correcta estructura termodinámica del problema. Así, se trata de un enfoque híbrido entre los métodos de caja negra basados en datos y las formulaciones analíticas tradicionales.

La primera parte de la tesis explora métodos de aprendizaje profundo que preservan la estructura para simular sistemas dinámicos a partir de datos. Esto se consigue utilizando varios sesgos inductivos, lo cual fuerza la estructura metripléctica de la dinámica por construcción y explota la estructura del dominio utilizando los principios del aprendizaje profundo geométrico. Estas arquitecturas de redes neuronales de última generación permiten identificar la dinámica de sistemas desconocidos incluso con un comportamiento altamente no lineal. Además, el esquema de integración resultante consigue predicciones rápidas y precisas con bajos requisitos de potencia computacional y almacenamiento en memoria.

La segunda parte describe un método para identificar la dimensionalidad intrínseca de un sistema dinámico con el fin de obtener un modelo de orden reducido, que

puede integrarse en el tiempo utilizando una arquitectura que preserva la estructura. Este método es conveniente en datos de alta dimensionalidad cuya compleja estructura resulta impráctica para las técnicas estándar.

La tercera y última parte explora dos posibles aplicaciones de los métodos expuestos en la tesis. Se describe un procedimiento de aprendizaje para sistemas abiertos en los que la interacción con fuentes externas es relevante, basado en el formalismo port-Hamiltoniano y su extensión a sistemas metriplécticos. La segunda aplicación es una *demo* de realidad aumentada para la interacción de sólidos deformables en tiempo real, posibilitada por las rápidas predicciones de los simuladores desarrollados en esta tesis.

Acknowledgements

There is a long list of marvellous people, which this section is too narrow to contain, that have inspired, encouraged or just supported me during these crazy 4 years of thesis.

I would like to start by thanking my supervisors, *Elías Cueto* and *Alberto Badías*. Thank you for giving me such an amazing opportunity, for all the guidance and blackboard sessions, and for the extremely valuable scientific and personal experiences that I have learnt from you. Thank you *Paco Chinesta*, for your advice and guidance since the very first paper of my thesis, as well as ESI Group for the financial support. I also have to mention my favourite mathematician of the department *David González* and my officemate *Icíar Alfaro*, I always could find a valuable advice from you. However, this thesis would not be possible without *Beatriz Moya*. She is the reason why I knocked at Elías' door and brings joy to the department, I would always be so grateful to her. Last but not least, thank you *Carlos Bermejo* and *Alicia Tierz* for putting up with me and my non-conformism, I hope that my advices and Github code will help you in your thesis journey.

Thanks to all my department colleagues: *José*, *Patri*, *Pau*, *Iulen*, *Itziar*, *Jacobo*, *Silvia*, *Nico* and *Diego*, for guiding and supporting me, specially in my early days in the thesis. After 6 months, the COVID-19 pandemic screwed things up with a historical lockdown. Miraculously, when I returned to the lab, I met *Álvaro*, *Ricardo*, *Benedetta*, *Elena*, *Raúl*, *Ángela* and *Marina*. Our "PhD Juepincho" group became inseparable, and all the travels, meetings and discussions about the PhD and life in general were amazing. Later came *Óscar*, *Pedro*, *Ángel* and *Elena*, great colleagues to continue the group activities and to whom I wish all the best in the academic world.

I want to thank Prof. *George Karniadakis* for giving me the opportunity to experience an amazing research intern at Brown University. It was a pleasure to learn from him and also from our collaborators, *Panos Stinis*, *Zhen Li* and *Zhen Zhang*. Also, thanks to all the people from the CRUNCH group, specially *Adar*, *Ahmad*, *Kamal*, *Sokratis*, *Mario* and *John*, for all the interesting talks about science and life in the US. Aside from the Applied Mathematics department, I also need to mention the "Beach

brunch" people: *Cris*, *Uriel*, *Sönke*, *Bjarne*, *Carlos*, *Raúl*, *Kass*, *Rocío* and *William*. It was incredible to travel, have parties and hang out with you all; you made those six months so worth it.

Thanks to all my ex-colleagues from the Graphics and Imaging Lab: *Manu*, *Vic*, *Ibón*, *Julio*, *Juan Raúl* and *Dario*. It is always very interesting to talk about advances in complementary research fields and share experiences from different points of view. Also, thanks to *José* from the robotics RoPeRT team, for those great beers discussing about music, science and mathematics since the beginning of our university days. Encouraged by the stimulating research topics of this thesis, I recently started a new academic journey in the mathematics field, in which I met some of the greatest minds of the new generations. *David J.*, *David T.*, *Pedro*, *Diego* and *Alonso*, it has always been fun to share our passion for Mathematics.

Aside from academic colleagues, I also want to mention my personal friends who have helped me to evade from the stressful research duties. The "Pishicas" *Marín*, *Babi*, *Cabrera* and *Eva*; the "Metalheads" *Chevy*, *Alas*, *Eli*, *Baquetas*, *Bassman*, *Cacho* and *Edu*; the "Lorem ipsum" *Álvaro*, *Isabel*, *Balma* and *Héctor*; the "Ingeniebrios" *Sethi*, *Carlos*, *Nacho*, *Ochoa*, *Peña* and *Alberto*; the "Desmadre" *Diego*, *Ingrid*, *Adrián*, *Pixa*, *Arturo*, *Alejandro*, *Carlos*, *Anita*, *Lorena*, *Álvaro* and *Luisito*. Thank you all for the great moments in concerts, house parties, and travels, which will always be with me. You are incredible.

This last thanks is very meaningful to me. None of this would be possible without the unconditional support of my whole family, specially my parents, *Marisol* and *Manolo*, who always believed in me and supported me in every decision of my life. All this work is for you.

Contents

List of Tables

List of Figures

List of Abbreviations

AE	**A**uto**E**ncoder
AI	**A**rtificial **I**ntelligence
AR	**A**ugmented **R**eality
CNN	**C**onvolutional **N**eural **N**etwork
CFD	**C**omputer **F**luid **D**ynamics
CPU	**C**entral **P**rocessing **U**nit
DPD	**D**issipative **P**article **D**ynamics
FEM	**F**inite **E**lement **M**ethod
FVM	**F**inite **V**olume **M**ethod
GAT	**G**raph **AT**tention Network
GCN	**G**raph **C**onvolutional **N**etwork
GNN	**G**raph **N**eural **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
GRU	**G**ated **R**ecurrent **U**nit
k-PCA	**k**ernel-**P**rincipal **C**omponent **A**nalysis
LLE	**L**ocally **L**inear **E**mbedding
LLM	**L**arge **L**anguage **M**odel
LSTM	**L**ong **S**hort-**T**erm **M**memory
MD	**M**olecular **D**ynamics
ML	**M**achine **L**earning
MOR	**M**odel **O**rder **R**eduction
MR	**M**ixed **R**eality
MSE	**M**ean **S**quared **E**error
MVP	**M**odel-**V**iew-**P**rojection
NN	**N**eural **N**etwork
ODE	**O**rdinary **D**ifferential **E**quation
PCA	**P**rincipal **C**omponent **A**nalysis
PDE	**P**artial **D**ifferential **E**quation
PINN	**P**hysics-**I**nformed **N**eural **N**etwork
POD	**P**rincipal **O**rthogonal **D**ecomposition

RNN	**R**ecurrent **N**eural **N**etwork
SAE	**S**parse **A**uto**E**ncoder
SPH	**S**moothed-**P**article **H**ydrodynamics
SPNN	**S**tructure-**P**reserving **N**eural **N**etwork
SSE	**S**um **S**quared **E**rror
SVM	**S**upport **V**ector **M**achine
TDA	**T**opological **D**ata **A**nalysis
TIGNN	**T**hermodynamics-**I**nformed **G**raph **N**eural **N**etwork
VR	**V**irtual **R**eality

Part I

Introduction and Background

1 Introduction

The study of physical dynamical systems has been of utmost importance in the early development of modern science. In fact, entire mathematical fields have risen as a consequence of the study of such disciplines. Differential calculus was motivated by the study of rigid body motions such as celestial bodies or ballistic studies. Fourier series were developed as a convenient method to solve the heat equation, and then generalized to integral transforms in harmonic analysis. The study of the relationship between heat and work on steam engines led to thermodynamics, a discipline which explains macroscopic behaviours in terms of microscopic constituents using statistical mechanics. These are only a few examples in which the discovery of complex physics phenomena led to the necessity of research on the convenient mathematical tools to describe them. As the famous Galileo Galilei once said,

'Mathematics is the language with which God has written the universe.'

In the engineering context, the mathematical description of physical systems is crucial in order to have a deeper understanding of their future behaviour and take the best cost-effective decisions in terms of design optimization. This is specially relevant in engineering fields with high operational costs, such as aeronautical and automotive industries, and can be achieved by the use of *computer simulations*. However, engineering systems usually consist of complex interacting geometries, so it is necessary not only to mathematically describe the process in terms of *ordinary* (ODEs) or *partial differential equations* (PDEs), but also to develop efficient numerical methods to solve them.

Although the methods presented in this thesis are formulated for arbitrary dynamical systems, we will focus on continuum mechanical systems: solid and fluid mechanics. Both disciplines are described by a set of PDEs derived from conservation laws such as mass, momentum or energy, resulting in the elasticity and the Navier-Stokes equations respectively. Even if these formulations are well known since the 19^{th} century, many research fields aiming to solve them optimally are still open. In the last decades, the gold standard methods have been the Finite Element Method

(FEM) in solid mechanics [253, 110] and the Finite Volume Method (FVM) in fluid mechanics [53], both consisting on representing and evaluating the continuum (infinite) formulation in a set of (finite) algebraic equations. Those methods have an extensive mathematical literature of stability and convergence theorems, which makes them reliable for the use in industry.

1.1 General Overview

With the irruption of the so-called fourth paradigm of science [101] a growing interest is detected in the machine learning of these scientific laws. A plethora of methods have been developed that are able to produce more or less accurate predictions about the response of physical systems in previously unseen situations by employing techniques ranging from classical regression to the most sophisticated deep learning methods. These data-driven procedures are sub-fields of the so-called *Artificial Intelligence* (AI).

Artificial intelligence can be defined as *"the study of agents that receive percepts from the environment and perform actions"* [228]. Although AI was already formally studied in the early 1950s, the field experienced several hype cycles followed by criticism and funding cuts, also known as AI winters [71]. However, recent advances in computational power, image processing techniques and Large Language Models (LLMs) among others have led to huge research interest in AI algorithms. This AI spring [273] has motivated the development and adaptation of models in other fields such as physics simulations, on which this thesis focuses.

This section explores some of the most important AI techniques, starting from classical Machine Learning to the more advanced tools in deep learning (see Fig. 1.1).

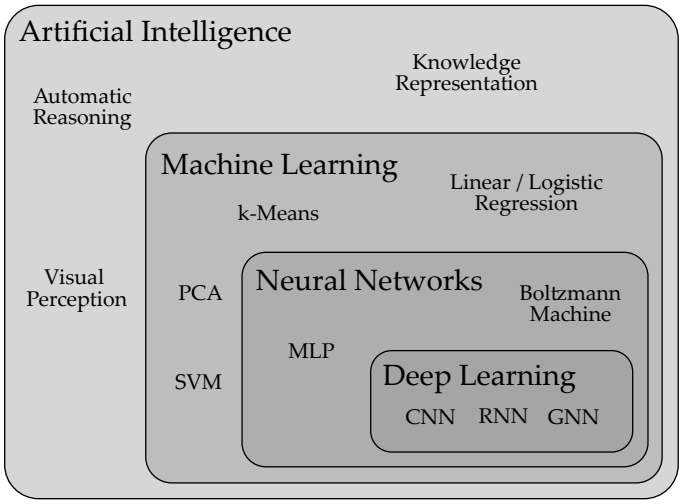


Figure 1.1: Hierarchical representation of Artificial Intelligence.

1.1.1 Machine Learning

Machine learning (ML) is a subset of AI algorithms which learn a certain problem automatically through experience by means of data. Thus, they are often referred to as *data-driven methods* and can be categorized into three types of algorithms. *Supervised* learning uses labelled data, which requires previous processing, whereas *unsupervised* learning is able to identify patterns in data without explicit labels. *Semi-supervised* learning is an intermediate approach which uses a small set of labelled data combined with a large amount of unlabelled data. Some important examples of such categories are the following:

- **Supervised**:
  - *Classification*: Labelling observations to classify into categories. Examples: Logistic Regression, Random Forest [102].
  - *Regression*: Estimation of the relationship among variables. Examples: Support Vector Machine (SVM) [40], Linear Regression.
- **Unsupervised**:
  - *Clustering*: Group data which share similarities. Examples: k-Means [90], k-Nearest Neighbour [60]
  - *Dimensionality reduction*: Reduction of arbitrary number of variables into principal (latent) variables. Examples: Principal Component Analysis (PCA) [270], Linear Discriminant Analysis (LDA) [38].
- **Semi-supervised**:
  - *Reinforcement leaning*: An agent takes an action in an environment to maximize a prescribed reward. Examples: Q-learning [37], Residual Algorithms [13].
  - *Generative models*: Learn complex data distributions and generate new samples. Examples: Mixture Models [276], Diffusion Models [241].

Among all the machine learning techniques briefly sketched here, there is one which stands out from the rest: the *Neural Network* model. It consists of artificial neuron units connected into layers, inspired by the biological neurons constituting animal brains. If the network has multiple layers it is usually referred to as *Deep Learning*, which enables to learn more complex representations of the studied problem in a hierarchical scheme.

1.1.2 Physics-Informed Deep Learning

A recent interest is observed in the incorporation of already existing scientific knowledge into these data-driven procedures, whose interest is twofold. Indeed, we prefer

not to get rid of centuries of scientific knowledge and rely exclusively on powerful machine learning strategies. Existing theories have proved to be useful in the prediction of physical phenomena and are still in the position of helping to produce very accurate predictions, as followed in the so-called *data-driven computational mechanics* [133]. On the other hand, these theories help to keep the consumption of data to a minimum. This is relevant because data are expensive to produce and maintain, so already existing scientific knowledge could alleviate the amount of data needed to produce a successful prediction.

This discipline has taken multiple names depending on the research group context: *physics-informed* machine learning [120, 121], *AI for Science* [262, 281], *scientific* machine learning [43], *physics augmented* learning [156] or *data-driven science and engineering* [25] to name a few. Each one has developed different approaches to enforce physical constraints to the deep learning algorithms, which we will refer to as *biases*. These biases can be categorized depending on the deep learning procedure in which they are applied:

1. **Database generation**: In supervised learning, simulation data from classical methods are usually used to generate a high-fidelity dataset which embodies the underlying physics of a phenomena. Another alternative is the use of measured data, in which several problems might occur such as sensor noise or impossibility to measure all the variables of the physical system. These are called *observational biases*.

2. **Architecture design**: The machine learning model can incorporate prior assumptions in terms of mathematical hard constraints called *inductive biases*. Those are related to the fulfilment of certain symmetries and conservation laws of the problem, which is the most principled way of obtaining a physical solution. Such approaches require sophisticated implementations which might be difficult to scale and optimize.

3. **Loss function**: The most popular approach to enforce physicality of the results is by including the governing equations in the loss function, which explicitly favours the convergence to solutions that satisfy those restrictions. Those penalty soft constraints are called *learning biases*. In this case, the constraints are only approximately satisfied and requires prior information about the governing dynamics, but provides a flexible and unsupervised framework.

This thesis will use a combination of the second and third points, i.e. enforcing physicality by construction in the architecture design together with soft constraints. For that purpose, it is necessary to dive into the mathematical foundations of inductive biases and the specific theoretical physics formulation used, which are addressed in Chapter 2.

1.2 Objectives

The main objective of this thesis is to develop deep learning algorithms to learn dynamical systems from data with the use of physics-based constraints. This objective is achieved by the consecution of the following subgoals:

- Design a learning procedure able to identify the correct metriplectic structure of high-fidelity data simulations.
- Develop model order reduction strategies to handle high-dimensional systems.
- Compare the existing methodologies and the proposed ones in terms of result accuracy and computational performance.
- Exploit the identified dynamics to create robust real-time simulators with thermodynamical guarantees.

1.3 Thesis Outline

This thesis is divided into five different parts. Part I consists of the first introductory Chapter 1, which explains the general overview, motivation and objectives of the thesis. Next in Chapter 2 a background and literature review of the relevant concepts of the thesis is presented. The main contributions of this thesis are presented in the next three parts:

- Part II presents two machine learning methods to learn thermodynamic consistent integrators of dissipative dynamical systems. Chapter 3 handles the problem using standard feed-forward neural networks in academic problems and Chapter 4 extends the algorithm to more complex unstructured domains using advanced tools based on geometric deep learning.
- Part III focuses on the study of structure-preserving model order reduction. In Chapter 5 we perform a reduction technique using sparse autoencoders which identify the intrinsic dimensionality of the problem in order to integrate it in time using structure-preserving neural networks.
- Part IV explores the performance of the developed techniques in several real-world applications. Chapter 6 aims to simulate complex systems by parts using port-Hamiltonian-like structures whereas Chapter 7 presents an augmented reality pipeline to simulate the interaction between the user and deformable virtual objects in real-time.

Last, Part V is dedicated to sum up all the conclusions of the thesis contributions and future work.

1.4 Contributions and Measurable Results

1.4.1 Publications

In the following section, we state the publications which support the contributions of this thesis. All the work presented has already been published in four top-tier journals indexed in the Journal Citations Reports (JCR) and SCIMAGO Journal and Country Rank (SJR). The specific publications are also explicitly described at the beginning of each chapter, with the thesis author underlined.

1. Q. Hernández, A. Badías, D. González, F. Chinesta, & E. Cueto
   Structure-preserving neural networks
   Journal of Computational Physics, 426, 109950 (2021)
   JCR Impact Factor: 4.645 (Q1 in Physics, Mathematical), SJR: 1.75 (Q1 in Applied Mathematics)

2. Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
   Deep learning of thermodynamics-aware reduced-order models from data
   Computer Methods in Applied Mechanics and Engineering, 379, 113763 (2021)
   JCR Impact Factor: 6.588 (Q1 in Engineering, Multidisciplinary), SJR: 2.22 (Q1 in Computational Mechanics)

3. Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
   Thermodynamics-informed graph neural networks
   IEEE Transactions on Artificial Intelligence, DOI: 10.1109/TAI.2022.3179681 (2022)
   JCR Impact factor: - (Not indexed yet), SJR: 1.38 (Q1 in Artificial Intelligence)

4. Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
   Port-metriplectic neural networks: thermodynamics-informed machine learning of complex physical systems
   Computational Mechanics, 72, 553–561 (2023)
   JCR Impact Factor: 4.391 (Q1 in Mathematics, Interdisciplinary Applications), SJR: 1.32 (Q1 in Applied Mathematics)

5. Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
   Thermodynamics-informed neural networks for physically realistic mixed reality
   Computer Methods in Applied Mechanics and Engineering, 407, 115912 (2023)
   JCR Impact Factor: 6.588 (Q1 in Engineering, Multidisciplinary), SJR: 2.22 (Q1 in Computational Mechanics)

1.4.2 Conference Communications

The contributions of the thesis have been communicated in several international conferences. The following list contains the oral and poster presentations given personally by the thesis author. Other communications given by co-authors about the present works are not included here.

Oral Presentations

1. WCCM-ECCOMAS 2020, Paris (France), Online
   Learning Physics Through Themodynamically-Informed Neural Networks
   14th World Congress in Computational Mechanics and *7th European Congress on Computational Methods in Applied Sciences and Engineering*

2. COUPLED 2021, Barcelona (Spain), Online
   Discovering the physical structure of dynamical systems with deep learning
   IX International Conference on Coupled Problems in Science and Engineering

3. YIC 2021, Valencia (Spain), Online
   Learning Physics through Themodynamically-Informed Neural Networks
   VI ECCOMAS Young Investigators Conference

4. MMLDT-CSET 2021, San Diego (United States), Online
   Simulating Physics with Structure-preserving Graph Neural Networks
   Mechanistic Machine Learning and Digital Twins for Computational Science, Engineering & Technology

5. SIAM-PD 2022, Berlin (Germany), Online
   Generic as an Inductive Bias in Neural Networks
   SIAM Conference on Analysis of Partial Differential Equations

6. WCCM-APCOM 2022, Yokohama (Japan), Online
   Learning physics with metriplectic and geometric biases
   15th World Congress in Computational Mechanics and *8th Asian Pacific Congress on Computational Mechanics*

7. ECCOMAS 2022, Oslo (Norway)
   Deep learning of dynamical systems using geometry and thermodynamics
   8th European Congress on Computational Methods in Applied Sciences and Engineering

8. CMN 2022, Las Palmas (Spain)
   Learning Mechanics with Geometry and Thermodynamics
   Congress on Numerical Methods in Engineering

9. GACM 2022, Essen (Germany)
   Learning Simulators with Geometry and Thermodynamics
   9th GACM Colloquium on Computational Mechanics

10. M2P 2023, Taormina (Italy)
    Deep learning of coupled dissipative systems
    Math 2 Product: Emerging Technologies in Computational Science for Industry, Sustainability and Innovation

11. YIC 2023, Porto (Portugal)
    AI-enhanced interactive simulators for virtual reality applications
    VII ECCOMAS Young Investigators Conference
12. ENUMATH 2023, Lisbon (Portugal)
    Structure-preserving neural networks for coupled dissipative systems
    European Conference on Numerical Mathematics and Advanced Applications
13. MMLDE-CSET 2023, El Paso (United States)
    Port-metriplectic neural networks for coupled dissipative systems
    2nd IACM Mechanistic Machine Learning and Digital Engineering for Computational Science Engineering and Technology

Poster Presentations

1. C2D3 Virtual Symposium 2020, Cambridge (United Kingdom), Online
   Learning physics with thermodynamically informed neural networks
   Cambridge Center for Data-Driven Discovery, Virtual Symposium, Research Rendezvous
2. ICLR 2021, SimDL Workshop, Vienna (Austria), Online
   Learned simulators that satisfy the laws of thermodynamics
   International Conference on Learning Representation 2021, Deep Learning for Simulation Workshop
3. IUTAM 2022, Paris (France), Online
   Learning simulators with geometry and thermodynamics
   IUTAM Symposium on Data-Driven Mechanics
4. LOG 2022, Boston (United States)
   Learning simulators with geometry and thermodynamics
   Learning on Graphs Conference 2022, MIT-CSAIL Meetup

1.4.3 Software

The results presented in this thesis can be replicated and/or extended using the publicly available code on Github `https://github.com/quercushernandez`. The repositories contain:

- High-fidelity simulation databases of the publication examples.
- Pre-trained model parameters and testing functions used to generate the results of the present thesis.
- Training functions to generate custom networks.
- Plotting and result statistics functions.

1.4.4 Research Internship

A 6 month research internship was carried out during the thesis.

- Supervisor: Prof. George Karniadakis.
- Dates: October 2022 - April 2023:
- Institution: Brown University, Providence (United States).

2 Background

This chapter summarizes the theoretical foundations on which this thesis will be constructed. In the first section, the basic building blocks of deep learning such as the Multilayer Perceptron and the Autoencoder are explored. The literature review is focused on the specific application to physics simulations and general differential equation solvers.

The second section focuses on more sophisticated models such as Convolutional, Recurrent and Graph Neural Networks formulated in terms of the geometric deep learning paradigm. The text is not intended to be exhaustive about all the possible deep learning architectures in the literature, but only the most relevant ones for the present thesis. More specific details of the theory and applications can be found in [76, 24].

The last section of the chapter explains the physical formalism used throughout the thesis. First, the canonical Hamiltonian formulation of classical mechanics is generalized to Poisson systems and finally to dissipative systems. More information about theoretical mechanics and non-equilibrium dynamics can be found in [7, 197].

2.1 The basics of Deep Learning

Machine learning is a discipline whose objective is to extract relevant information from data. From the mathematical point of view, machine learning not only aims to learn the (probably non-linear) mapping between inputs and outputs but also the representation itself. This is usually referred to as *representation learning*. Deep learning is a smaller field in machine learning that tackles this problem by introducing representations that are expressed in terms of other simpler representations, called *features*. This is achieved with the composition of several learning blocks whose combination is able to learn more complex patterns from the previous ones. It becomes very obvious in computer vision, where an abstract concept (content of an image)

is decomposed as a combination of simpler patterns (edges, corners) with different degrees of complexity, see Fig. 2.1.

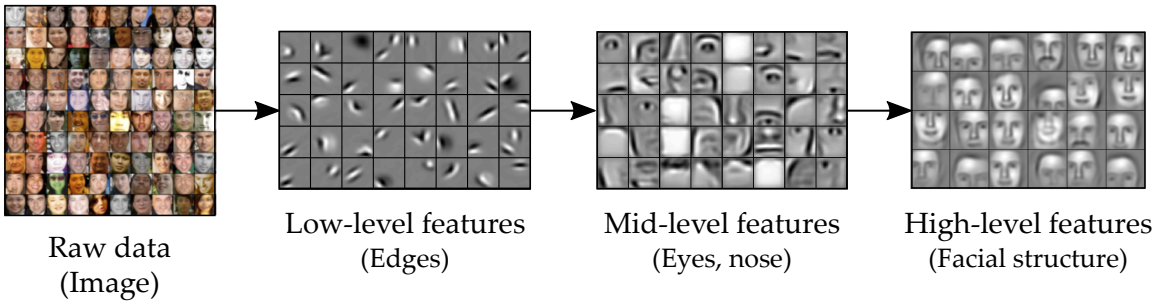


Figure 2.1: Representation learning in face recognition. By using deep learning, each layer adds an extra level of complexity to the hidden representations or features. At the end, the network is able to recognize objects present in the image as a combination of its simpler parts. Source of images: [142].

In fact, deep learning revolutionized image processing as it offered a consistent way to mathematically parametrize a human perception, such as concepts in an image, into hierarchical features suitable for optimize via minimization techniques. This was possible with the development of Convolutional Neural Networks, which offered an ideal framework specially engineered for image data structures. Similarly, special architectures were designed for other data structures such as time series (Recurrent Neural Networks) or natural language texts (Transformers). Physics problems have also an intrinsic mathematical structure unveiled by centuries of theoretical and experimental knowledge. From Newton's laws of motion to the Standard Model, the laws of nature can often be described as a set of partial differential equations (PDEs) which can also lead neural networks to find the correct solution of a dynamical problem. These are the foundations of the so-called *physics-informed deep learning*, as already introduced in Chapter 1.

2.1.1 Multilayer Perceptron

The simplest building block of artificial deep neural network architectures is the neuron or perceptron (Fig. 2.2, left). Several neurons are stacked in a *multilayer perceptron* (MLP), which is mathematically defined as follows

$$\boldsymbol{x}^{[l]} = \sigma(\boldsymbol{W}^{[l]}\boldsymbol{x}^{[l-1]} + \boldsymbol{b}^{[l]}), \tag{2.1}$$

where l is the index of the current layer, $\boldsymbol{x}^{[l-1]}$ and $\boldsymbol{x}^{[l]}$ are the layer input and output vector respectively, $\boldsymbol{W}^{[l]}$ is the weight matrix of the last layer, $\boldsymbol{b}^{[l]}$ is the bias vector of the last layer and σ is the activation function. If no activation function is applied, the MLP is equivalent to an affine transformation. However, σ is chosen to be a nonlinear function in order to increase the capacity of modelling more complex problems,

which are commonly nonlinear. When fully assembled, the MLP transforms an input feature vector $\boldsymbol{x} \in \mathbb{R}^{N_{\text{in}}}$ into an output vector $\boldsymbol{y} \in \mathbb{R}^{N_{\text{out}}}$.

$$\begin{aligned} \text{MLP}_{\phi} : \mathbb{R}^{N_{\text{in}}} &\longrightarrow \mathbb{R}^{N_{\text{out}}}, \\ \boldsymbol{x} &\longmapsto \boldsymbol{y}. \end{aligned}$$

where $\phi = \{\boldsymbol{W}, \boldsymbol{b}\}$ represent the trainable parameters, i.e. weights and biases, of the MLP.

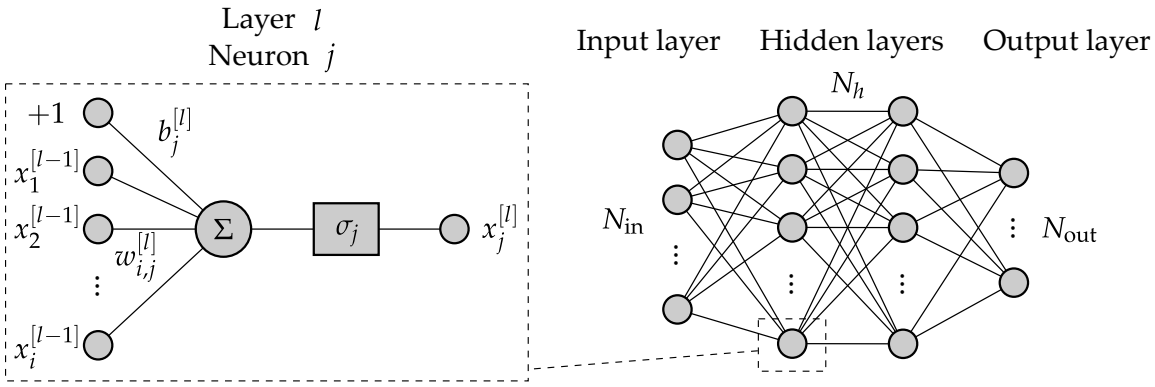


Figure 2.2: Representation of a single neuron (left) as a part of a fully connected neural net (right). The complete architecture is commonly referred to as multilayer perceptron or fully-connected neural network.

The main reason under the fact that neural networks are able to learn and reproduce such a variety of problems is that they are considered to be universal approximators [44, 106], meaning that they are capable of approximating any measurable function to any desired degree of accuracy.[1]

Standard MLPs has been widely used in physical simulations for their simplicity and versatility. The most popular approach to physical priors in machine learning are the so-called *Physics-informed Neural Networks* (PINNs) [214, 120]. The idea is to make use of MLPs to learn the solution of a partial differential equation (PDE) by minimizing its residual in predefined collocation points subject to the initial and boundary conditions. Incorporating the residual equation to the learning procedure acts as a regularizer in the learning process [191] enhancing its generalization capability [168], see Fig. 2.3. There is an extense literature using PINNs applied to different engineering fields such as fluid mechanics [117, 28, 218], solid mechanics [86, 216, 85, 84], material science [154, 20, 87], heat transfer [193] to name a few.

Recent work in *Deep Operator Networks* (DeepONets) [158, 159] extends the notion of PINNs to operators, enabling to learn parametric problems with no re-training for a

[1] The original proof takes various strong assumptions, such as the use of squashing (logistic-like) activation functions and compact domains. Several versions of the universal approximation theorem were developed in subsequent works relaxing the original assumptions [147, 209, 126]

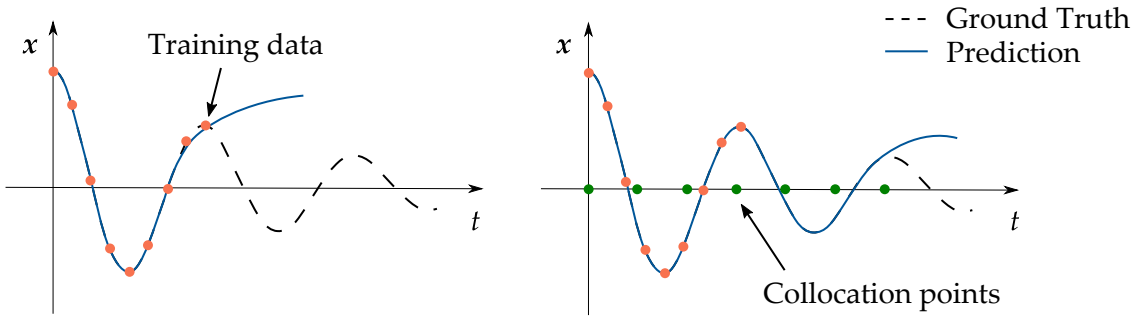


Figure 2.3: Neural network fitting model without (left) and with physics priors (right). The incorporation of the residual of a PDE equation in the collocation points acts as a regularization and data augmentation in the learning process whereas in a classical data-driven model fails to capture the dynamics outside of the training/interpolating domain.

new set of parameters [266]. The architecture consist of two subnetworks: a brunch whose input are the sensors of the problem (input function) and a trunk to evaluate the output function. This idea has been also exploited in several works such as multiphysics problems [169, 27], heat transport [160], geology [115], wave mechanics [146], etc. Multilayer perceptrons have also being used to create differentiable simulators for robotic applications [212, 94] and contact dynamics [208].

2.1.2 Autoencoders

An *autoencoder* [76] is a type of artificial neural network which modifies the dimensionality of an input into a coded version, which ideally contains the same information, by learning the identity function. It is composed by an *encoder* E_ϕ, which maps input data $\boldsymbol{x} \in \mathbb{R}^{N_{\text{in}}}$ onto a code or *latent vector* $\boldsymbol{y} \in \mathbb{R}^{N_{\text{code}}}$, and a *decoder* D_θ, which applies the inverse mapping back to the original space,

$$\mathrm{E}_\phi : \mathbb{R}^{N_{\text{in}}} \longrightarrow \mathbb{R}^{N_{\text{code}}} \qquad \mathrm{D}_\theta : \mathbb{R}^{N_{\text{code}}} \longrightarrow \mathbb{R}^{N_{\text{in}}}$$
$$\boldsymbol{x} \longmapsto \boldsymbol{y}. \qquad \boldsymbol{y} \longmapsto \boldsymbol{x}.$$

Usually, autoencoders are used to generate reduced order models of high-dimensional data in a bottleneck structure ($N_{\text{code}} < N_{\text{in}}$) called *undercomplete* autoencoder. Conversely, *overcomplete* autoencoders expand the input dimension to generate a higher feature latent space ($N_{\text{code}} > N_{\text{in}}$).

The vanilla autoencoder scheme, where the encoder and decoder are modelled with standard MLPs, has been used in many fields such as physics [54, 272, 59], chemistry [153], mechanics [144, 93] or computer graphics [236]. They have been used to discover representations of the Koopman eigenfunctions from data [244, 162, 201]. *Fourier Neural Operator* [152] uses an overcomplete encoder to decompose the input physical variables into its Fourier components. In the field of learnt simulators, autoencoder-based model order reduction has been used to learn differentiable

physics simulators for control purposes [17].

Variational autoencoders have been widely used in generative models and time series modelling. The most successful architecture is *Neural Ordinary Differential Equations* [32, 45, 124, 127], which make use of ODE solvers as a black-box block inside the deep learning pipeline to predict the latent variables of a time series.

2.2 Geometric Deep Learning

Pure MLPs-based architectures are agnostic to the data structure of the studied problem. This is, the input features are considered to have no preferred relationships between each other, as all the neurons are fully-connected (see Fig. 2.2). Even if that data can be processed with vanilla MLPs, specific architectures were developed to exploit that data structures like Convolutional Neural Networks [141] in image processing and Recurrent Neural Networks [227] in time series processing. Both architectures exploit the structured data in grid elements and timesteps to induce translation and time invariances respectively, see Fig. 2.4, and are considered as major breakthroughs in their respective fields. The reason why those architectures succeeded is because these models present strong *inductive biases* [14], which are learning priors induced by construction in the learning process which drives the learning algorithm to certain desirable solutions (Fig. 2.5).

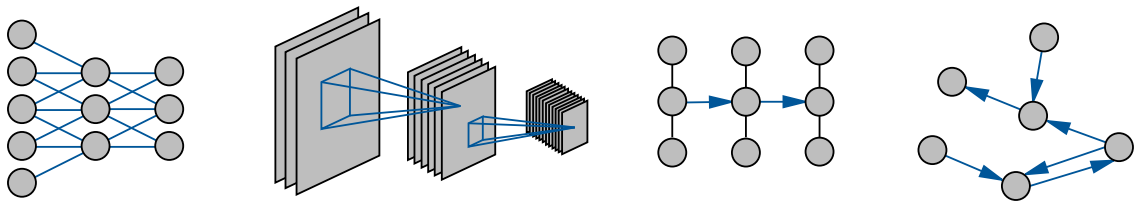


Figure 2.4: Various relational inductive biases in standard deep learning network architectures.

Motivated by the study of such architectures, a new machine learning paradigm recently arose known as *geometric deep learning* [23]. The main insight is that, even if big data problems suffer from the curse of dimensionality [109], most tasks of interest have predefined regularities arising from an underlying low-dimensional mathematical structure. The main objective of geometric deep learning is to expose those regularities through unified geometric principles that can be applied to a wide spectrum of applications. This concept is already widely used in physics, where the identification of the symmetries of the problem is fundamental to formulate new theories [161]. In deep learning, those symmetries were also exploit in popular traditional architectures already mentioned in Fig. 2.4 but were not treated as a unified

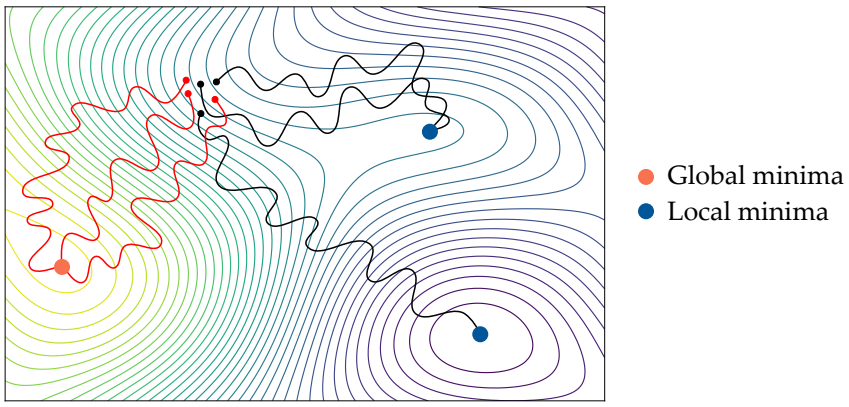


Figure 2.5: Effect of inductive biases in a loss landscape. A model with weak inductive biases (black lines) can be equally attracted to several local minima [178], and the converged solution can be arbitrarily affected by random variations [243]. A model with strong inductive biases (red lines) is driven by construction to particular solutions, in this case the global minima.

framework. In the next section, a brief introduction to geometric deep learning will be presented based on [24].

2.2.1 Group Theory

A *symmetry* of an object is a transformation that leaves a certain property of that object invariant. Symmetries are inherent in many machine learning tasks, like translational symmetry in image processing or time warping in time series prediction. The set of symmetries of an object has a particular algebraic structure known as a group. Thus, the central mathematical model for studying geometric deep learning is based on group theory.

We first need to clarify the distinction between the mathematical space where the concepts of group theory are formulated, denoted as *domain* Ω, and the vector space of the output features, denoted as $\mathcal{C}$ where $\dim(\mathcal{C})$ are the *channels*. The relationship between the domain and the channel space is called a *signal* $x : \Omega \to \mathcal{C}$. The set of all possible signals are gathered in the *signal space* $\mathcal{X}(\Omega) = \{x : \Omega \to \mathcal{C}\}$. The input, output and feature vectors used in all the machine learning applications explained below are contained on this high-dimensional space. A familiar example is an RGB image, depicted in Fig. 2.6. Symmetry concepts might not be obvious in the signal domain $\mathcal{X}(\Omega)$, but have been widely studied in the mathematical abstract domain Ω. The objective is to translate domain concepts to the real-world signal domain.

Now we are ready to start with a few basic definitions of group theory. A *group* is a set G that together with an internal operation $\star : G \times G \to G$ satisfies that for all $g, h, k \in G$ we have:

- Associativity: $(g \star h) \star k = g \star (h \star k)$

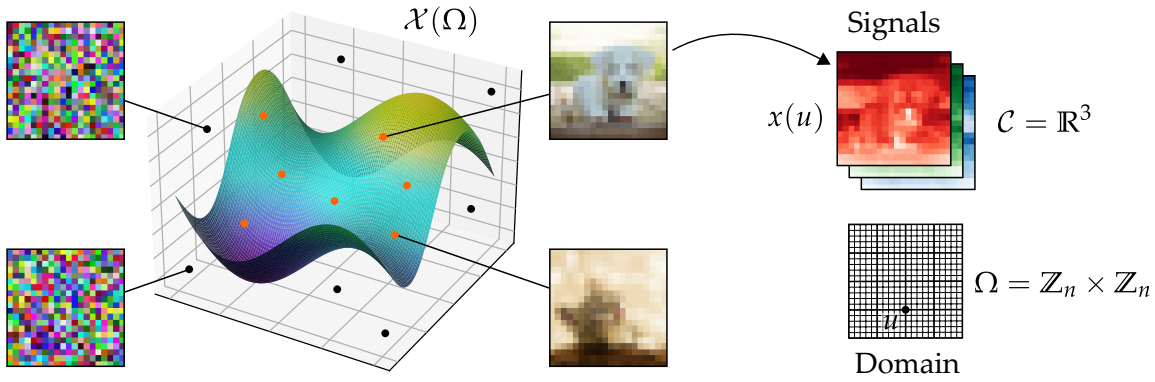


Figure 2.6: Example of the signal space $\mathcal{X}(\Omega)$ of 2D images and its abstract representation in a domain Ω. The signal space contains all the possible $n \times n$ RGB images, which is very high-dimensional. However, an image can also be considered as a signal $x(u)$ of a domain $u \in \Omega$. The advantage of that identification is that you can now exploit the the underlying low-dimensional structure of $\mathcal{X}(\Omega)$ by studying the simpler domain Ω.

- Identity: Exists a unique $e \in G$ satisfying $e \star g = g \star e = g$
- Inverse: Exists a unique g^{-1} satisfying $g \star g^{-1} = g^{-1} \star g = e$

Fig. 2.7 depicts the group induced by the symmetries of a triangle, also called the dihedral group D_3, and contains all the rotations and reflections that maintain the triangle unchanged. This is an example of a *discrete* or *finite group*.

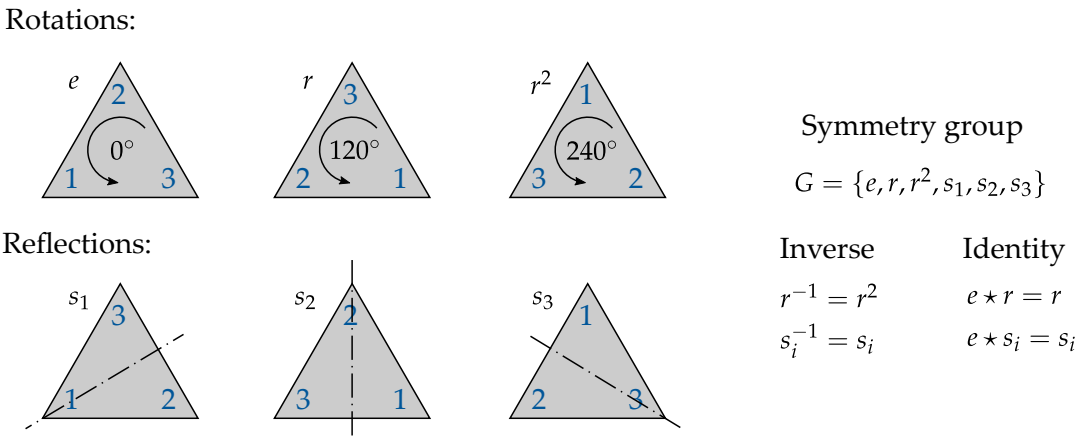


Figure 2.7: Example of the dihedral discrete group D_3. The symmetries of a triangle define a set of 6 transformations which meet all the requirements to be considered a (discrete) group.

Groups can also be continuous. A very useful *continuous group* is the set of all invertible $n \times n$ matrices together with the usual matrix multiplication, which is called the general linear group $\mathrm{GL}(n) = \{\boldsymbol{M} \in \mathbb{R}^{n \times n} \,|\, \det(\boldsymbol{M}) \neq 0\}$. It represents the symmetry of linear transformations that preserve the vector space structure, i.e. vector addition and scalar multiplication. If that group is restricted to orthogonal matrices with determinant 1, it is called special orthogonal group $\mathrm{SO}(n) = \{\boldsymbol{M} \in \mathbb{R}^{n \times n} \,|\, \boldsymbol{M}^\top = \boldsymbol{M}^{-1}, \det(\boldsymbol{M}) = 1\}$, which encodes all the possible rotational symmetries of an n-sphere and models rotation transformations in $\mathbb{R}^n$.

Note that commutativity is not required for the definition of a group. If the group is commutative, i.e. $g \star h = h \star g$, the group is called *Abelian*. An example of Abelian group is a vector space V with the sum of vectors as internal operation $(V, +)$ which is called an additive group. The definition of group is very simple yet powerful, as a priori very different kind of objects and transformations might be represented by the same abstract group.

The interesting property about groups is that you can define an action over a domain Ω and study how entities in Ω transform. A *(left) group action* $\Phi : G \times \Omega \to \Omega$ is a mapping such that for all $u \in \Omega$ and $g, h \in G$ satisfies the following properties:

- Identity: $\Phi(e, u) = u$ for $e \in G$ the group identity.
- Compatibility: $\Phi(g \star h, u) = \Phi(g, \Phi(h, u))$

The notation $\Phi(g, u)$ will be shortened as $g \cdot u$ for the rest of the text. For example, in the case of the Euclidean group in $\mathbb{R}^2$ denoted as $\mathrm{E}(2)$ you can define actions that preserves Euclidean distances, which consist of translations, rotations and reflections. The same group can act on the space of images, by translating, rotating and flipping the image pixels. This is, by defining an action on a group $G = \mathrm{E}(2)$ over a low-dimensional domain $\Omega = \mathbb{R}^2$ automatically induces an equivalent action on the high-dimensional signal space $\mathcal{X}(\Omega)$ of images, see Fig. 2.8. This new action has the following expression

$$(g \cdot x)(u) = x(g^{-1} \cdot u), \tag{2.2}$$

where $g \in G$, $u \in \Omega$ and $x(u) \in \mathcal{X}(\Omega)$. One can check that it is a valid group action, as it satisfies associativity. Note that for simplicity we denote again the group action over the signal space with the same "$\cdot$" symbol as the domain space. However, they are two conceptually different operations.

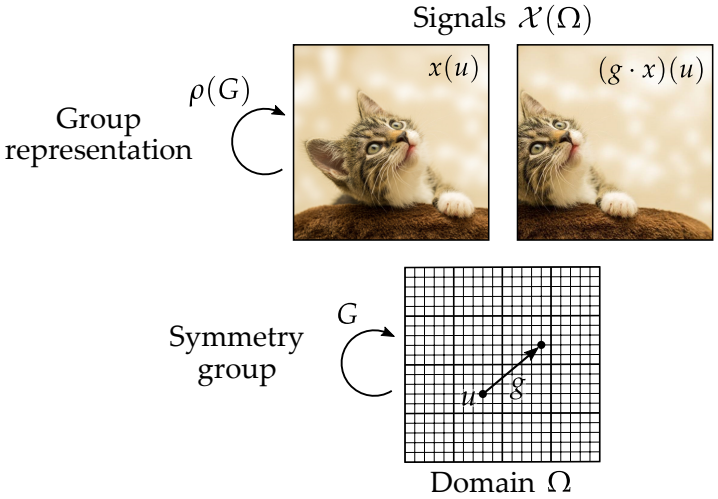


Figure 2.8: Group action $\Phi(g, u) = g \cdot u \in \Omega$ over a domain and its equivalent action $(g \cdot x)(u) \in \mathcal{X}(\Omega)$ in the signal space.

A particular case of group actions is when the mapping Φ is linear, which is called *group representation*. These actions are specially important as they allow us to describe linear actions by using invertible matrices. This is, a group representation is a map $\rho : G \to \mathrm{GL}(n)$ that assigns to each group element $g \in G$ an invertible $n \times n$ matrix $\rho(g) \in \mathrm{GL}(n)$. Similarly to the definition of group action, the group internal operation and the matrix multiplication must be consistent. This is, for all $g, h \in G$:

- Compatibility: $\rho(g \star h) = \rho(g)\rho(h)$

Now Eq. (2.2) can be transformed into the usual matrix multiplication

$$(g \cdot x)(u) = \rho(g)x(u).$$

In the Euclidean group example, the distance-preserving transformations can be represented as a composition of translation, rotation and reflection matrices. Those transformations are called *isometries* and are widely used in robotics and computer vision.

Now we are ready to apply the abstract group theory concepts to machine learning. We have seen how a group G can act on some domain Ω to transform its entities in a specific way. We also saw that if that transformation is linear, it can be transferred to the signal domain $\mathcal{X}(\Omega)$ via its matrix representation. As we discussed in the introduction, the symmetries of Ω underlying the signals can impose a structure to the functions defined over the signal domain. Two important cases are invariant and equivariant functions. A function $f : \mathcal{X}(\Omega) \to \mathcal{Y}$ is

- *G-invariant* if $f(\rho(g)x) = f(x)$
- *G-equivariant* if $f(\rho(g)x) = \rho(g)f(x)$

for all $x \in \mathcal{X}(\Omega)$ and $g \in G$. This is, invariant functions leave the output unaffected by the group action, whereas equivariant functions transform the output the same way as the inputs, see Fig. 2.9. This is a powerful inductive bias, as it reduces the space of possible learnable functions to achieve a learning task.

However, the geometric priors described so far with theory group do not provide a specific architecture for building such representation, but rather a series of necessary conditions. In practice, the basic building blocks of geometric deep learning can be summarized in the following architectures:

- *Equivariant layers*: Operations $E : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$ such that E is G-equivariant.
- *Non-linearity*: Element-wise non-linear transformation $\sigma : \mathcal{C} \to \mathcal{C}'$ such that $(\sigma(x))(u) = \sigma(x(u))$.
- *Local pooling layers*: Coarsening layers $\phi : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C})$ which downsamples the domain, i.e. $\Omega' \subseteq \Omega$.

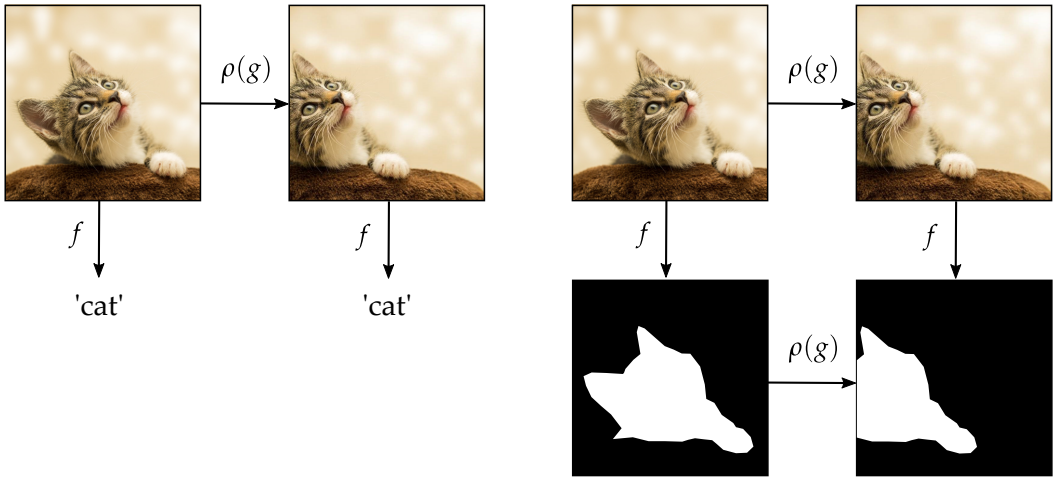


Figure 2.9: Invariant (left) and equivariant (right) functions under a shift transformation. Image classification tasks are invariant to the orientation of the objects whereas image segmentation tasks are equivariant.

- *Global pooling layers*: Operations $\psi : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$ such that ψ is G-invariant.

To sum up, a general G-invariant function $f : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$ can be constructed with a composition of the previous operations

$$f = \psi \circ \phi_i \circ \sigma_i \circ E_i \cdots \phi_1 \circ \sigma_1 \circ E_1,$$

where the domain and codomain of each subsequent operation match. Depending on the input and output space particular features, each of those layers can be engineered with the appropriate layer architecture. Over the past decades, a vast amount of deep learning architectures exploited such symmetries for different signal spaces (Table 2.1). The following sections will explore the basic concepts behind each architecture, with references in physical simulations application.

Table 2.1: Several deep learning architectures with possible symmetry groups.

Architecture	**Domain** Ω	**Symmetry group** G
RNN	1D Grid	Time Warping W
CNN	2D Grid	Translation $T(2)$
Spherical CNN	Sphere	Rotation SO(3)
GNN	Graph	Permutation Σ_n
Transformer	Complete Graph	Permutation Σ_n

2.2.2 Convolutional Neural Networks

Vanilla CNNs [140] were first widely used in image processing and then modified with many variants, such as the Convolutional Autoencoders [172], Generative Adversarial Networks (GAN) [77] or Fully Convolutional Networks [157]. A *convolution* of a signal x defined over a group G with a filter K is defined as

$$(x * K)(g) = \int_{u \in G} x(u) K(g^{-1} \cdot u) d\mu(u),$$

where $g \in G$ and $d\mu(u)$ is the measure of integration. In the case of the usual image convolution, over the translation group $G = T(2)$ we can get the discretized version of the convolution[2] as a sliding kernel $\boldsymbol{K}$ of width W_k and height H_k over the image $\boldsymbol{I}$ pixel coordinates (u, v):

$$(\boldsymbol{I} * \boldsymbol{K})_{u,v} = \sum_{i=1}^{H_k} \sum_{j=1}^{W_k} K_{i,j} I_{u+i,v+j}.$$

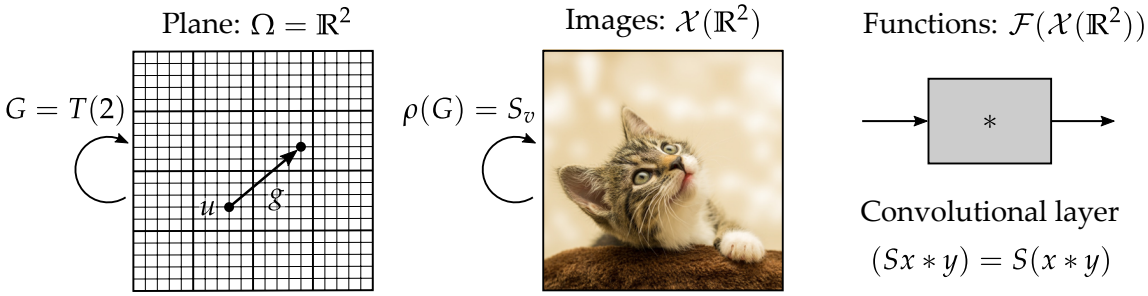


Figure 2.10: Convolutional Neural Networks in terms of group representations. The domain has an underlying symmetry group over the translation group $T(2)$, whose representation in the signal space is the shift operator S_v. CNNs exploit the mentioned symmetry using convolutional layers which are translational equivariant.

Convolutional neural networks have been widely explored in fluid mechanics [248, 75, 195, 260, 155, 242], as physical phenomena described in Eulerian coordinates have similar structure of an image. However, they are also widely used in fields like solid mechanics [174, 263], high energy physics [211, 33] or heat transport [283, 164]. In the context of learnt simulators, [134] use the combination of CNNs and MLPs for perception and prediction respectively.

2.2.3 Recurrent Neural Networks

The classical approach to process time series are recurrent neural networks. The time correlations in RNNs are learnt by using a cyclic connection between the hidden

[2]The convolution operation used in CNNs is a misnomer. In practice, convolutions are implemented as cross-correlations, which are equivalent to convolutions with rotated filters.

states $\boldsymbol{h}(t)$ of the neural network, allowing to identify temporal patterns of data snapshots $\boldsymbol{z}(t)$. The most popular recurrent architectures are Bidirectional RNN [234], Long Short-Term Memory (LSTM) networks [103] and Gated Recurrent Unit (GRU) [35].

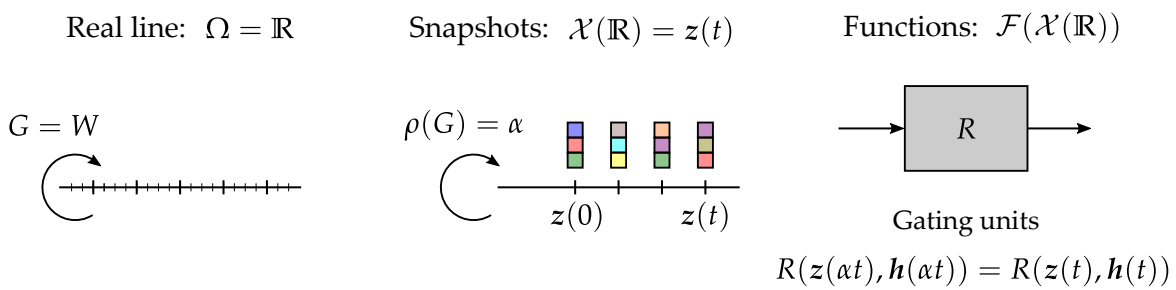


Figure 2.11: Gating Recurrent Neural Networks in terms of group representations. The domain has an underlying symmetry group over the warping group W, whose representation in the signal space is a time warping mapping[3](in this case, a time rescaling). The gating mechanisms in RNNs such as LSTM networks make the architecture invariant to time transformations [245], such as irregular sampling of the temporal data.

Many examples of works can be found in the literature of time series forecasting [252, 12], nonlinear dynamics [179, 229], solid [67, 280] and fluid mechanics [118, 149], heat transport [166], etc. Recurrent neural networks have also been used in learn simulators for contact trajectories [2, 56].

2.2.4 Graph Neural Networks

More recently, the use of graph neural networks (GNN) has revolutionized previous approaches with very promising results. A clear example are Transformer models [257], which make use of attention to overcome the limited short-term memory of RNNs.

The basic data structure used in GNNs is the *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{u})$ where $\mathcal{V} = \{1, ..., n\}$ is a set of $|\mathcal{V}|$ *vertices*, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of $|\mathcal{E}|$ *edges* and $\boldsymbol{u}$ are the *global features*. The vertices represent entities of the problem and the edges indicate the (directed) relationship between them. These relationships are encoded by the *adjacency matrix* $\boldsymbol{A} = \{a_{ij} = 1 \text{ if } (i,j) \in \mathcal{E} \text{ else } 0\}$.

The graph features are processed following some prescribed rules. In a general graph, we can define the *vertex feature vector* $\boldsymbol{v}_i \in \mathbb{R}^{F_v}$, the *edge feature vector* $\boldsymbol{e}_{ij} \in \mathbb{R}^{F_e}$ and the *global feature vector* $\boldsymbol{u} \in \mathbb{R}^{F_g}$. The feature dimensions are F_v, F_e and F_g respectively. The transformation from an initial graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{u})$ to a processed graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \boldsymbol{u}')$ can be performed with several generic operations based on [14] (see Fig. 2.12), which can be summed up in the following:

[3]Time transformations in the form of $\tau : \mathbb{R}^+ \to \mathbb{R}^+$ for any monotonically increasing differentiable mapping τ.

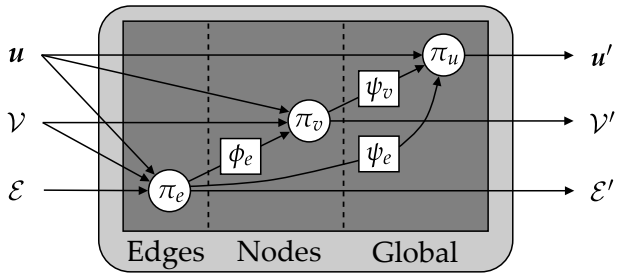


Figure 2.12: General processing block of a Graph Neural Network.

- **Edge block**: The updated edge features $\boldsymbol{e}'_{ij}$ are computed as a function π_e of the previous edge features, the sending and receiving nodes based on the adjacency matrix and the global features. These edge-wise features are generally called *messages* from node i to node j

$$\begin{aligned} \pi_e : \mathbb{R}^{F_e+2F_v+F_g} &\longrightarrow \mathbb{R}^{F_e} \\ (\boldsymbol{e}_{ij}, \boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{u}) &\longmapsto \boldsymbol{e}'_{ij}, \end{aligned}$$

  where $(\cdot, \cdot)$ denotes vector concatenation.

- **Node block**: For each node i the messages are pooled with a (local) aggregation function ϕ_e based on the *neighborhood* $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$ of the node. Then, the updated node features $\boldsymbol{x}'_i$ are computed with a second processing function π_v using the current node features, the pooled messages and the global features. This gathering operation is called the *message passing algorithm*.

$$\begin{aligned} \pi_v : \mathbb{R}^{F_v+F_e+F_g} &\longrightarrow \mathbb{R}^{F_v} \\ (\boldsymbol{v}_i, \phi_e(\boldsymbol{e}'_{ij}), \boldsymbol{u}) &\longmapsto \boldsymbol{v}'_i. \end{aligned}$$

- **Global block**: The global feature vector is updated with the collapsed information of all the edges and vertices of the graph, by using two (global) aggregation functions ψ_v and ψ_u over all the edges $(i,j) \in \mathcal{E}$ and vertices $i \in \mathcal{V}$ respectively,

$$\begin{aligned} \pi_u : \mathbb{R}^{F_v+F_e+F_g} &\longrightarrow \mathbb{R}^{F_u} \\ (\psi_v(\boldsymbol{v}'_i), \psi_u(\boldsymbol{e}'_{ij}), \boldsymbol{u}) &\longmapsto \boldsymbol{u}'. \end{aligned}$$

The processing functions π_e, π_v and π_u can be modelled as standard MLPs, which are shared along all vertices and edges. The aggregation functions ϕ_e, ψ_v and ψ_u are chosen to be commutative in order to achieve permutation equivariance. This is, the solution is permuted the same under a permutation $\boldsymbol{P}$ of the initial features $\boldsymbol{v}_i = \boldsymbol{X}$, see Fig. 2.13.

This general GNN scheme can be simplified to get all the successful architectures found in the literature. The most used models are Graph Convolutional Networks

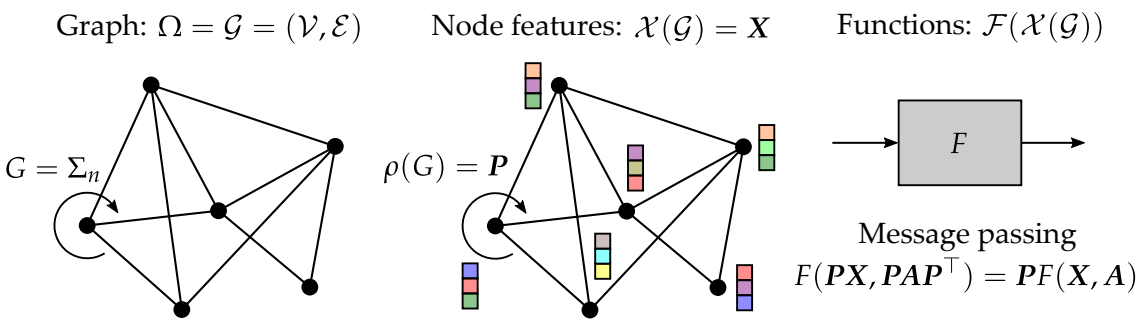


Figure 2.13: Graph Neural Networks in terms of group representations. The domain has an underlying symmetry group over the permutation group Σ_n, whose representation in the signal space is a permutation matrix $\boldsymbol{P}$. The message passing algorithm in GNNs is permutation equivariant, as the aggregation functions used are commutative functions.

(GCN) [131], Graph Attention Networks (GAT) [259], GraphSAGE [88] and Message Passing Neural Networks [69].

Many works used the versatility of GNNs for predicting physical simulations in unstructured domains. Several graph-based works have achieved great improvements in physics problems such as predicting atomic bond forces [107, 39], particle tracking in high energy physics [238, 119], n-body problem with more general interactions [130, 14, 41], or learning simulators to predict complex fluid interactions [231] and meshed domains [207, 284]. Graph-based neural networks have been also extensively used for differentiable simulators [150, 16, 4].

2.3 Structure of Dynamical Systems

As already pointed out in the last section, the development of new physic theories relies on identifying the underlying symmetries of the problem [161]. This is a direct consequence of *Noether's (first) theorem*, which states that every differentiable symmetry of a physical system has a corresponding conservation principle. Translational and rotation invariances are related to the conservation of linear and angular momentum respectively, whereas time translation invariance implies the conservation of energy. This theorem establishes a direct bridge between the exploit of the symmetries of a problem, the basic principle of geometric deep learning, and the fulfilment of conservation laws, required for robust and realistic physical simulations. This section explores the mathematical foundations of classical physics formulations and their extension to dissipative dynamics[4] in order to incorporate those physical priors to deep learning algorithms.

The field of theoretical mechanics has studied dynamic phenomena with the starting point of Newton's laws of motion in the 18$^{\text{th}}$ century. From then, many authors

[4]Noether's theorem was originally formulated for conservative phenomena. Its extension to dissipative systems might not necessarily hold [64].

continued to study those equations, giving rise to more modern and sophisticated reformulations. The two dominant branches of theoretical mechanics are Lagrangian mechanics and Hamiltonian mechanics, which are equivalent formulations to describe the dynamics of a conservative system (see Table 2.2).

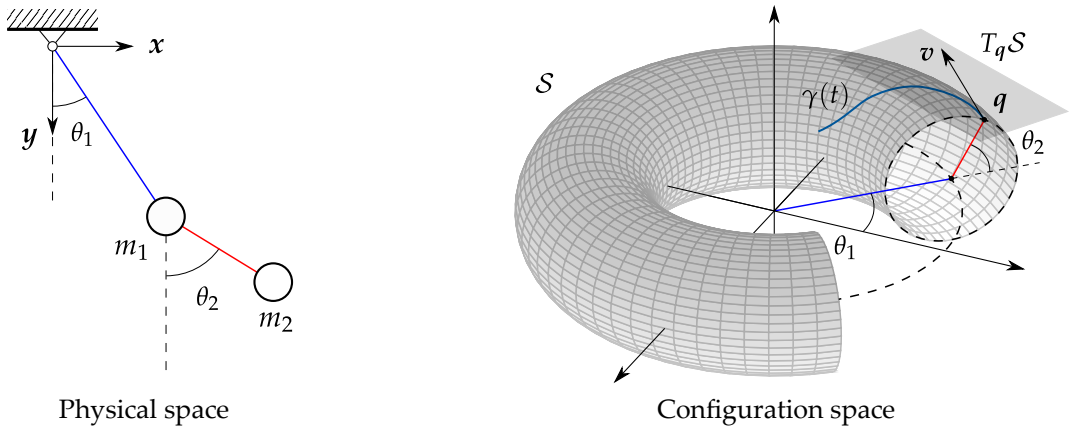
Table 2.2: Classical mechanics formalisms for conservative systems.

	Newtonian	**Lagrangian**	**Hamiltonian**
Equation	Newton's laws	Euler-Lagrange	Hamilton
Quantity	Forces	Lagrangian	Hamiltonian

The next sections develop a progressively general framework to handle more complicated systems, starting with the classical Hamiltonian formalism, extended to Poisson systems and finally dissipative dynamics.

2.3.1 Hamiltonian Dynamics

Consider a collection of N particles, defined in space by the generalized coordinates or *position* $\boldsymbol{q} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_N)$. The position vector lies in the *configuration space* $\mathcal{S}$, which is a manifold that contains every possible configuration that those particles can occupy based on the physical restrictions of the system (see Fig. 2.14). As the configuration space is a differentiable manifold, we can define the tangent space of $\mathcal{S}$ at $\boldsymbol{q}$, denoted as $T_{\boldsymbol{q}}\mathcal{S}$, which contains the *velocities* of the particles $\boldsymbol{v} = \dot{\boldsymbol{q}} = (\dot{\boldsymbol{q}}_1, ..., \dot{\boldsymbol{q}}_N)$ for any given trajectory $\gamma(t)$ passing through $\boldsymbol{q}$.



Physical space Configuration space

Figure 2.14: Physical and configuration space of a 2D double pendulum. The position of the two masses is determined by $\theta_1, \theta_2 \in [0, 2\pi)$, each one representing a circumference S^1 in the physical space $\mathbb{R}^2$. Then, the configuration space is a manifold described by a torus $\mathcal{S} = T^2 = S^1 \times S^1$, with points of coordinates $\boldsymbol{q} = (\theta_1, \theta_2)$ and velocities $\boldsymbol{v} = \dot{\boldsymbol{q}} = (\dot{\theta}_1, \dot{\theta}_2)$.

The Hamiltonian paradigm is based on the consideration of two independent variables: the position and the generalized *momentum* $\boldsymbol{p} = (\boldsymbol{p}_1, ..., \boldsymbol{p}_N)$. The momentum is defined as the gradient of the Lagrangian with respect to the velocities in

Lagrangian mechanics, so it lies on the dual space of $T_{\boldsymbol{q}}\mathcal{S}$: the *cotangent space* $T^*_{\boldsymbol{q}}\mathcal{S}$. Both quantities represent the *phase space* of the problem, and it is naturally defined as the *cotangent bundle* of the physical space $T^*\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{p}) \mid \boldsymbol{q} \in \mathcal{S}, \boldsymbol{p} \in T^*_{\boldsymbol{q}}\mathcal{S}\}$, which encodes all possible states of the system (Fig. 2.15).

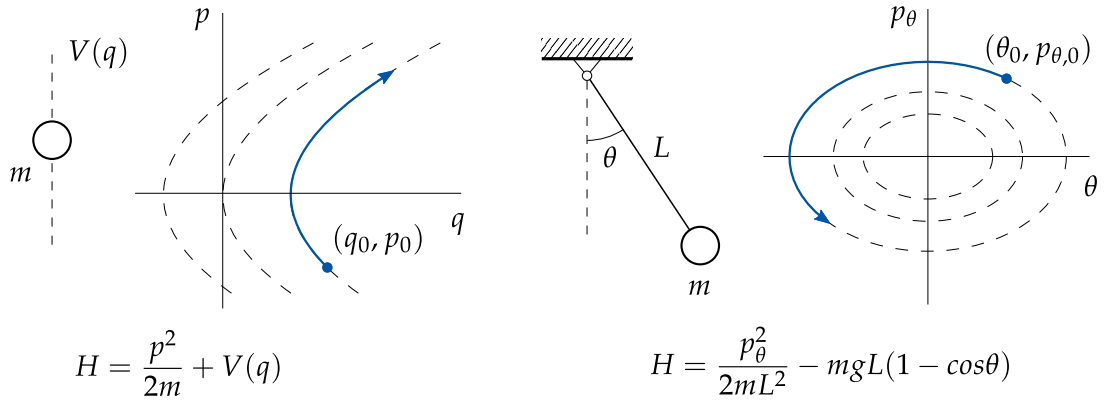


Figure 2.15: Phase space diagrams of a particle in a potential and a simple pendulum. In the particle potential, the coordinates are (q, p) whereas in the pendulum case (θ, p_θ) where $p_\theta = mL^2\dot{\theta}$. In these cases, the Hamiltonian is equivalent to the total energy of the system, which is composed by the kinetic $K(\boldsymbol{p})$ and potential $V(\boldsymbol{q})$ energy.

Denoting by $H(\boldsymbol{q}, \boldsymbol{p}) : T^*\mathcal{S} \to \mathbb{R}$ the *Hamiltonian* of the system,[5] the time evolution of both variables is given by the Hamilton's canonical equations of motion

$$\frac{d\boldsymbol{q}}{dt} = \frac{\partial H}{\partial \boldsymbol{p}}, \quad \frac{d\boldsymbol{p}}{dt} = -\frac{\partial H}{\partial \boldsymbol{q}}. \tag{2.3}$$

They express the reversible kinematics of the phase space variables $\boldsymbol{q}$ and $\boldsymbol{p}$, meaning that the system is invariant under time translations. By recalling Noether's theorem, one can translate this time symmetry into an energy conservation law, which can be easily derived using the chain rule

$$\frac{dH}{dt} = \frac{\partial H}{\partial \boldsymbol{q}} \cdot \frac{d\boldsymbol{q}}{dt} + \frac{\partial H}{\partial \boldsymbol{p}} \cdot \frac{d\boldsymbol{p}}{dt} = \frac{\partial H}{\partial \boldsymbol{q}} \cdot \frac{\partial H}{\partial \boldsymbol{p}} - \frac{\partial H}{\partial \boldsymbol{p}} \cdot \frac{\partial H}{\partial \boldsymbol{q}} = 0.$$

Motivated by the symmetry of Eq. (2.3) and the importance of Hamilton's equations in physics, the mathematical structure of those systems has been studied in detail into a formulation called Poisson algebra. This will be very useful in order to generalize to non-canonical and dissipative systems.

Poisson algebra

Suppose that we are interested in the time evolution of some function of the phase space variables and time $f(\boldsymbol{q}, \boldsymbol{p}, t) : T^*\mathcal{S} \to \mathbb{R}$ usually referred to as an *observable*.

[5] The Hamiltonian of a system can be derived directly from Lagrangian mechanics by considering the Legendre transformation between the conjugate variables $(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ and $(\boldsymbol{q}, \boldsymbol{p}, t)$.

The total time derivative of f is

$$\frac{df}{dt} = \frac{\partial f}{\partial \boldsymbol{q}} \cdot \frac{d\boldsymbol{q}}{dt} + \frac{\partial f}{\partial \boldsymbol{p}} \cdot \frac{d\boldsymbol{p}}{dt} + \frac{\partial f}{\partial t}.$$

By using the Hamiltonian mechanics relationship in Eq. (2.3) we get the following expression

$$\frac{df}{dt} = \frac{\partial f}{\partial \boldsymbol{q}} \cdot \frac{\partial H}{\partial \boldsymbol{p}} - \frac{\partial f}{\partial \boldsymbol{p}} \cdot \frac{\partial H}{\partial \boldsymbol{q}} + \frac{\partial f}{\partial t}.$$

We now define the binary operation called the *Poisson bracket* $\{\cdot,\cdot\} : T^*\mathcal{S} \times T^*\mathcal{S} \to T\mathcal{S}$ of two arbitrary observables $g(\boldsymbol{q},\boldsymbol{p},t)$ and $h(\boldsymbol{q},\boldsymbol{p},t)$ as

$$\{g,h\} = \frac{\partial g}{\partial \boldsymbol{q}} \cdot \frac{\partial h}{\partial \boldsymbol{p}} - \frac{\partial g}{\partial \boldsymbol{p}} \cdot \frac{\partial h}{\partial \boldsymbol{q}} \tag{2.4}$$

we can finally establish the following compact formula

$$\frac{df}{dt} = \{f,H\} + \frac{\partial f}{\partial t}. \tag{2.5}$$

In particular, we can write Hamilton's equations as Poisson bracket relations considering that $\boldsymbol{q}$ and $\boldsymbol{p}$ do not have an explicit time dependence

$$\frac{d\boldsymbol{q}}{dt} = \{\boldsymbol{q},H\}, \quad \frac{d\boldsymbol{p}}{dt} = \{\boldsymbol{p},H\}.$$

The Poisson bracket operation has a number of important properties. Formally, the set of the phase space observables and the binary operation of a Poisson bracket result in a *Lie algebra* structure. Thus, for any observables f,g,h and $a,b \in \mathbb{R}$ we have

- (Bi)linearity: $\{af+bg,h\} = a\{f,h\} + b\{g,h\}$
- Skew-symmetry: $\{f,g\} = -\{g,f\}$
- Jacobi identity: $\{f,\{g,h\}\} + \{h,\{f,g\}\} + \{g,\{h,f\}\} = 0$

Using the skew-symmetry property, we can derive also that $\{f,f\} = 0$. In addition to the Lie algebra structure, the Poisson bracket also acts as a derivative operator, which gives an additional associativity property:

- Product (Leibniz) rule: $\{fg,h\} = f\{g,h\} + g\{f,h\}$

It becomes obvious that the symmetric role of position and momentum is a key feature in Hamiltonian dynamics and more general Poisson structures. Thus, we can define a unified notation into a single phase space variable vector $\boldsymbol{z} = (\boldsymbol{q},\boldsymbol{p})$ and a

skew-symmetric block matrix $\boldsymbol{J}$, transforming Eq. (2.3) in a more compact expression

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{J}\frac{\partial H}{\partial \boldsymbol{z}}, \quad \boldsymbol{J} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{pmatrix}. \tag{2.6}$$

Likewise, in this notation the Poisson bracket of Eq. (2.4) takes a very simple form

$$\{g, h\} = \frac{\partial g}{\partial \boldsymbol{z}} \cdot \boldsymbol{J}\frac{\partial h}{\partial \boldsymbol{z}}. \tag{2.7}$$

From the differential geometry perspective, the skew-symmetric condition of $\boldsymbol{J}$ and the smooth manifold generated by the phase space $T^*\mathcal{S}$ of the system result in a so-called *symplectic manifold* structure [8, 83]. Consequently, the matrix $\boldsymbol{J}$ is often called the *canonical symplectic matrix*. In the next section we extend the notion of Poisson algebra to model non-canonical and dissipative phenomena.

Poisson systems

The first step is to use the mathematical properties of the Poisson brackets to overcome the limitation of canonical variables[6] in the Hamiltonian paradigm. Lets consider a set of state variables $\boldsymbol{z} \in T^*\mathcal{S}$, which may not be canonical, and a more general type of skew-symmetric matrix $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{z})$. We can reformulate Eq. (2.6) as

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{L}\frac{\partial H}{\partial \boldsymbol{z}}. \tag{2.8}$$

Similarly, the Poisson braket in matrix form of Eq. (2.7) transforms to

$$\{g, h\} = \frac{\partial g}{\partial \boldsymbol{z}} \cdot \boldsymbol{L}\frac{\partial h}{\partial \boldsymbol{z}}.$$

Now the system is called a *Poisson system* and $\boldsymbol{L}$ is called non-canonical symplectic matrix or *Poisson matrix*, and in general depends on the state variables $\boldsymbol{z}$. From now, we drop this explicit dependence to reduce the notation. The canonical formulation just remains as a special case where $\boldsymbol{L} = \boldsymbol{J}$ is composed of identity blocks.

By defining the non-canonical Hamiltonian equations, a new set of constants of motion apart from the Hamiltonian arise that depend on the degeneracy of the Poisson bracket. These set of non-trivial functions are called *Casimir functions* $C = C(\boldsymbol{z})$, for which the Poisson bracket with any other observable $f(\boldsymbol{q}, \boldsymbol{p}, t)$ is zero

$$\{C, f\} = \boldsymbol{0}, \quad \forall f(\boldsymbol{q}, \boldsymbol{p}, t).$$

In particular, if we consider the Hamiltonian we get $\dot{C} = \{C, H\} = \boldsymbol{0}$, so the Casimirs are a constant of motion. In other words, if we modify the actual Hamiltonian of the

[6] They are called canonical because they satisfy Eq. (2.3), i.e. Hamilton's canonical equations.

system by adding a linear combination of Casimirs $H' = H + \lambda_i C_i$, it generates the same dynamics as H. Note that the existance of Casimirs implies that $\boldsymbol{L}$ is not full rank, and those Casimirs lie on the kernel[7] of $\boldsymbol{L}$. In that case, the Poisson bracket is called to be *degenerate*.

2.3.2 Dissipative Dynamics

The study of dissipative systems in engineering is crucial to develop trustworthy simulations of the real world. Very few phenomena in nature is perfectly conservative. Every real mechanical system is subject to friction forces, which are responsible for the imperfect transfer of energy, wasted in the form of heat. Similar effects arise in many other disciplines, such as electrical power dissipation in conducting media, irreversible processes in thermodynamics or viscous dissipation in fluid mechanics. Experimental data measured from real systems will unlikely follow the conservative equations of the Hamiltonian paradigm. Thus, the development of theoretical extensions of the later to dissipative systems is of utmost importance in the study of machine learning applied to engineering systems.

As we already saw in previous sections, the Hamiltonian formalism is based on the conservation of the Hamiltonian and can be formulated with a rich mathematical structure called the Poisson algebra. However, systems that are non conservative would not a priori be expected to fit into de form of Eq. (2.8). Furthermore, it is not clear which quantity remains constant for time translation invariance (energy conservation) and which binary bracket is convenient for a rich mathematical structure. The following section is based on the theoretical framework of the *GENERIC formalism* and *metriplectic systems*, which were an independent development of Morrison [184, 183], Grmela and Öttinger [78, 200] and Kaufman [122].

Lets consider the *state space* $\mathcal{M} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{p}, \ldots) \mid \boldsymbol{q} \in \mathcal{S}, \boldsymbol{p} \in T^*_{\boldsymbol{q}}\mathcal{S}, \ldots\}$ defined as the usual phase space coordinates $(\boldsymbol{q}, \boldsymbol{p})$, which are not required to be canonical, in addition to other variables considered to define the energetic state of the system up to a certain level of detail.[8] From now on, the vector $\boldsymbol{z}$ will be referred to as the *state vector* of the system. The main idea is to consider a *generalized free energy* $\mathcal{F} : \mathcal{M} \to \mathbb{R}$ defined as

$$\mathcal{F} = E + S,$$

where $E : \mathcal{M} \to \mathbb{R}$ is the *energy* and $S : \mathcal{M} \to \mathbb{R}$ is called the *entropy*, and is defined to be an arbitrary function of the Casimirs of the Poisson bracket. This is inspired on the energy formulation of thermodynamics, in which the equilibrium state is obtained

[7] In the case of canonical Hamiltonian systems the matrix $\boldsymbol{L} = \boldsymbol{J}$ is always full rank, i.e. the bracket is non-degenerate.

[8] The original formulation clearly states that there is no preferred universal set of state variables. There are indeed many different choices of the state variables such that their respective brackets replicate the dynamics of the state variables $\boldsymbol{z}$ [200].

by extremizing the energy at constant entropy.[9] In analogy with the Poisson algebra, we also need a binary bracket in order to produce the equations of motion. Thus we define a dissipative generalization of Eq. (2.5) with the angle brace notation

$$\frac{df}{dt} = \langle f, \mathcal{F} \rangle + \frac{\partial f}{\partial t} = \{f, \mathcal{F}\} + [f, \mathcal{F}] + \frac{\partial f}{\partial t}.$$

Here we can see the reason to choose $\mathcal{F}$ for generating the dynamics of the problem, as the critical points of the generalized energy $\nabla \mathcal{F} = 0$ correspond to dynamical equilibria $\dot{f} = 0$. Note that the generalized bracket $\langle \cdot, \cdot \rangle : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ has been split into a skew-symmetric $\{\cdot, \cdot\}$ and a symmetric contribution $[\cdot, \cdot]$. The first is the usual skew-symmetric Poisson bracket and the second is a new symmetric bracket which has similar properties of a *metric tensor*. This is, for any functions f, g, h of the state space and time and $a, b \in \mathbb{R}$ the bracket satisfies

- (Bi)linearity: $[af + bg, h] = a[f, h] + b[g, h]$
- Symmetry: $[f, g] = [g, f]$
- Non-Negativity: $[f, f] \geq 0$

In matrix notation, the bracket can be expressed as

$$[g, h] = \frac{\partial g}{\partial z} \cdot \boldsymbol{M} \frac{\partial h}{\partial z},$$

where $\boldsymbol{M}$ is a symmetric and positive-semidefinite matrix called the *friction matrix*.

Finally, assuming that there is no explicit time dependence, applying the bilinearity property of the brackets, using the Casimir and Entropy definition $\{\cdot, S\} = 0$ and adding an extra degeneracy condition on the new bracket $[\cdot, E] = 0$, we get that for a set of state variables z, their time evolution is governed by the following expression

$$\frac{dz}{dt} = \{z, E\} + [z, S]. \tag{2.9}$$

A more practical and familiar expression using the matrix notation reads

$$\frac{dz}{dt} = \boldsymbol{L} \frac{\partial E}{\partial z} + \boldsymbol{M} \frac{\partial S}{\partial z}. \tag{2.10}$$

This is called the *General Equation for Non-linear Equilibrium of Reversible-Irreversible Coupling* (GENERIC) and it is of utmost importance in the development of the present thesis. We must remember that for constructing Eq. (2.10) we needed to consider the

[9] For instance, we could consider any function of the form $\mathcal{F}_\lambda = E + \lambda S$ where λ is a Lagrange multiplier of the minimization problem [185]. A particular case of this is the Helmholtz free energy in thermodynamics, which is defined as $F = U - TS$ where U is the internal energy, T is the temperature and S is the entropy.

Casimir restrictions, which can be expressed in matrix form as

$$\boldsymbol{M}\frac{\partial E}{\partial z} = \boldsymbol{L}\frac{\partial S}{\partial z} = \boldsymbol{0},$$

known as the *degeneracy conditions*. We can check that Eq. (2.10) has by construction all the desirable properties:

- The energy potential function E plays the role of the Hamiltonian H in pure Hamiltonian or Poisson systems. It is conserved, as inherits all the desirable properties from the Poisson bracket, thus it satisfies the first law of thermodynamics (energy conservation):

$$\frac{dE}{dt} = \{E, E\} + [E, S] = \frac{\partial E}{\partial z} \cdot \boldsymbol{L}\frac{\partial E}{\partial z} + \frac{\partial E}{\partial z} \cdot \boldsymbol{M}\frac{\partial S}{\partial z} = 0.$$

- The entropy potential function S represents the counterpart of the energy in dissipative dynamics. The loss of energy from the dissipative dynamics is balanced with the entropy generation and has independence with respect to the conservative dynamics, as it is defined to be a Casimir of the Poisson bracket. It can be easily derived that it satisfies the second law of thermodynamics (entropy inequality):

$$\frac{dS}{dt} = \{S, E\} + [S, S] = \frac{\partial S}{\partial z} \cdot \boldsymbol{L}\frac{\partial E}{\partial z} + \frac{\partial S}{\partial z} \cdot \boldsymbol{M}\frac{\partial S}{\partial z} = \frac{\partial S}{\partial z} \cdot \boldsymbol{M}\frac{\partial S}{\partial z} \geq 0.$$

A pure dissipative system described by the dissipative bracket is called a *metric system*, as the matrix $\boldsymbol{M}$ plays the role of a metric tensor. The combination of both conservative (symplectic) and dissipative (metric) dynamics gives rise to the term *metriplectic system*, as depicted in Fig. 2.16.

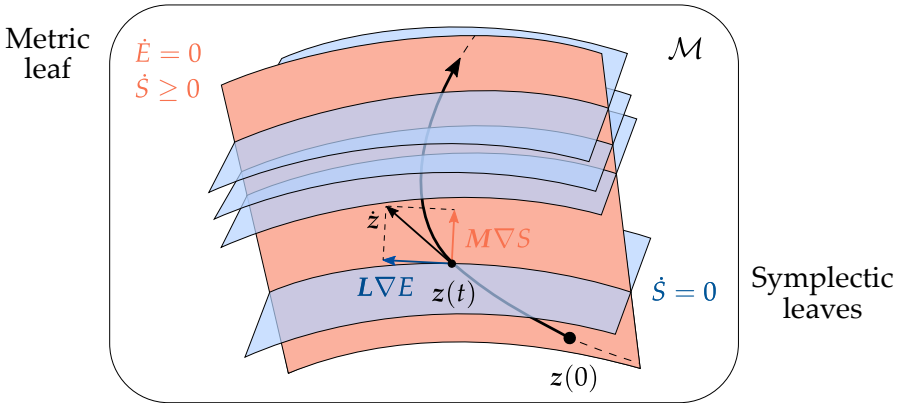


Figure 2.16: Metriplectic phase space $\mathcal{M}$ for a general dissipative dynamical system. The conservative dynamics lie inside the symplectic leaves (blue) represented as level sets of constant entropy $\dot{S} = 0$, showing the reversibility of Hamiltonian dynamics. The GENERIC dynamics take place in a single metric leaf (orange), i.e. surfaces with constant energy $\dot{E} = 0$. Conceptualization from [180].

Many examples are found within the bibliography which formulate the GENERIC structure of a wide variety of physical phenomena. The original formulation aimed at the description of complex rheological models such as polymeric fluids and reptation models [200], as well as both classical and extended irreversible thermodynamics [80]. Independently, the compressible Navier-Stokes equations [185] and the Euler's equation of rigid bodies [183, 173] were also shown to be metriplectic. Several applications in plasma physics models can be found such as the Vlasov-Poisson equation [274], dissipative relativistic plasmas [123], Lorenz-covariant dissipative systems [254] and the smoothed dissipative particle dynamics model [52]. Recent works also explore the metriplectic formulation of dissipative thermoelasticity [221, 220, 175].

Multiscale problem

One of the advantages of the GENERIC formalism is that it offers a structure which is applicable to any *level of description* of the system (Fig. 2.17). This coarse-graining procedure is widely used in thermodynamics and statistical physics [51].

- **Microscale**: In the microscale, the position and momentum of all the particles of the system are known, so there is full information of the atomic interactions and the Hamiltonian principles hold. In terms of the GENERIC formalism, there is only a conservative term with no dissipation ($\boldsymbol{L} \neq \boldsymbol{0}$, $\boldsymbol{M} = \boldsymbol{0}$). This scale is rarely used in computational mechanics due to the huge amount of data needed for the description of the state of the system.
- **Mesoscale**: The mesoscale covers a wide variety of granularity levels, from the molecular scale to crystal structures. The single atomic details are neglected in favour of a simplified version of "pseudo-atoms", which drastically reduces the degrees of freedom of the system easing the simulation of such structures. These unresolved degrees of freedom are precisely manifested in entropy generation ($\boldsymbol{L} \neq \boldsymbol{0}$, $\boldsymbol{M} \neq \boldsymbol{0}$) via the fluctuation-dissipation theorem [137, 235].
- **Macroscale**: At this coarser scale the atomic particles have completely vanished, and the conditions of continuum mechanics can be considered. Most of engineering computational modelling lies on this level of description.
- **Thermodynamics**: In classical thermodynamics, the complex microscopic dynamics of a system are averaged to a few measurable macroscopic quantities, based on statistical mechanics. This is the coarsest level of detail, in which only invariant quantities are used.

An equivalent approach is the Mori-Zwanzig projection formalism [182, 288], which provides a mathematical framework to construct coarse-graining models from microscale highly-detailed simulations.

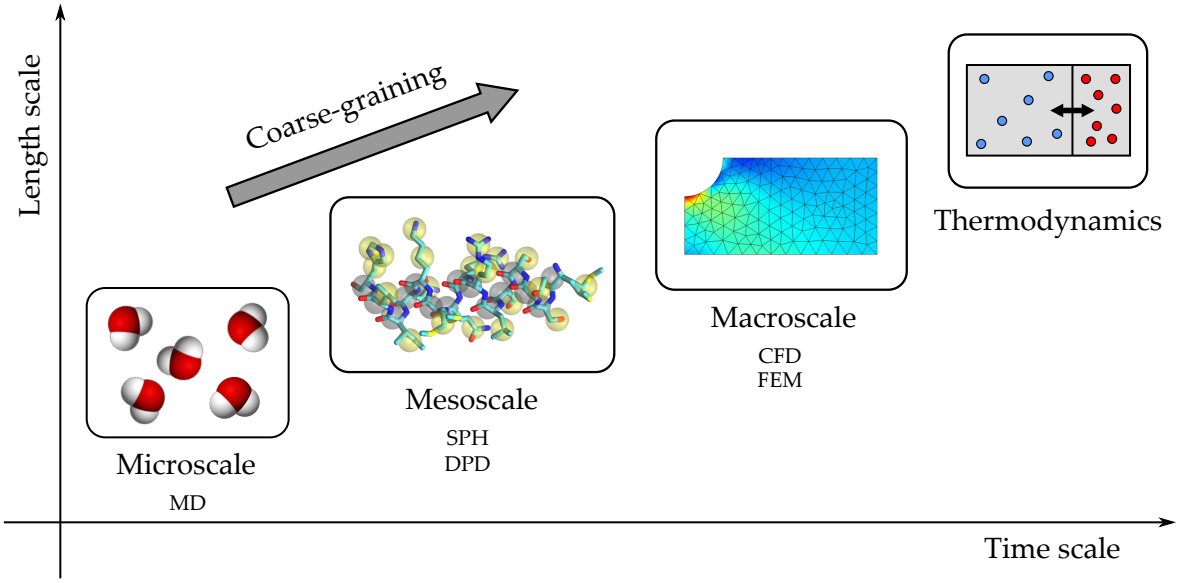


Figure 2.17: Different levels of description of a dynamical system. The coarse-graining process simplifies complex systems at various levels, reducing the data needed to represent the system but losing information of the details.

To sum up, the GENERIC formalism not only provides a thermodynamically consistent structure to handle general dynamical systems, but also a model valid for different levels of granularity of a simulation.

2.3.3 Structure-Preserving Deep Learning

The rich mathematical structure of symplectic and metriplectic systems is a strong inductive bias to learn the underlying physics of a problem. In this case, the problem is to identify the potential functionals (Hamiltonian, Lagrangian, etc) based on the state variables, becoming a system identification problem. Thus, the only information available to the deep learning procedure is a basic physical structure that the dynamics must follow, but no ad hoc equations are explicitly imposed as in previous physics-informed algorithms. We refer to this new approach as *structure-preserving deep learning*.

Several authors take advantage of the Hamiltonian structure to construct symplectic integrators to predict conservative dynamical systems [230, 116, 176, 249, 34]. Others, use the Hamiltonian principles to design more expressive deep neural network architectures [63] or to find the Hamiltonian function and phase space from data [18, 251]. The Hamiltonian paradigm is also widely used in quantum mechanics, where similar deep learning literature can be found in problems such as electron dynamics [21], learning ground states [135] or optimal control [66]. Alternative formulations can be developed by resorting to the equivalent Lagrangian formalism, see [22, 163, 287, 145, 5], among others.

The present thesis main contribution is the extension those methods, only valid for conservative systems, to dissipative systems via the GENERIC formalism. Following this work, several alternative algorithms have later been proposed including the degeneracy conditions as hard constraints using tensor factorization [143] or skew-symmetric matrices [282].

In the context of model order reduction, several reduction techniques have been developed to conserve the Hamiltonian [206, 70, 1, 165, 100] and metriplectic [81] structure of the full order systems. Alternative approaches to dissipative systems have later been explored through the *Onsager principle* [277, 108].

Part II

Deep Learning of Dynamical Systems

Structure-Preserving Neural Networks

3

In this chapter, we develop a method to learn physical systems from data by using feedforward neural networks and whose predictions comply with the first and second principles of thermodynamics. The method employs a minimum amount of data by enforcing the metriplectic structure of dissipative Hamiltonian systems in the form of the so-called General Equation for the Non-Equilibrium Reversible-Irreversible Coupling, GENERIC. The method does not need to enforce any kind of balance equation, and thus no previous knowledge on the nature of the system is needed. Conservation of energy and dissipation of entropy in the prediction of previously unseen situations arise as a natural by-product of the structure of the method. Examples of the performance of the method are shown that comprise conservative as well as dissipative systems, discrete as well as continuous ones. The content of this chapter is included in the following publication [95]:

Q. Hernández, A. Badías, D. González, F. Chinesta, & E. Cueto
Structure-preserving neural networks
Journal of Computational Physics, 426, 109950 (2021)

3.1 Introduction

As already pointed out in Chapter 1, there is a growing interest in the machine learning of scientific laws. For instance, recent works in solid mechanics have substituted the constitutive equations with experimental data [133, 9], while conserving the traditional approach on physical laws with high epistemic value (i.e., balance equations, equilibrium), also known as data-driven computational mechanics. Similar approaches have applied this concept to the unveiling (or correction) of plasticity models [112], while others created the new concept of constitutive manifold [113, 114]. Other approaches are designed to unveil an explicit, closed form expression for the physical law governing the phenomenon at hand [26].

In the context of deep learning, many approaches have already been explored in the last years to solve PDEs with neural networks, as presented in Section 2.1. Additionally, several works aim to exploit the symplectic structure of Hamiltonian dynamics to learn conservative systems, see Section 2.3. However, many real engineering systems, thus measured data, do not follow the conservative model due to dissipative effects. For that reason, it is relevant to develop machine learning tools to learn such systems using the convenient inductive biases.

Several co-authors have introduced the so-called *thermodynamically consistent* data-driven computational mechanics [72, 73, 68]. Unlike other existing works, this approach does not impose any particular balance equation to solve for. Instead, it relies on the imposition of the right thermodynamic structure of the resulting predictions, as dictated by the GENERIC formalism [78]. This ensures conservation of energy and the right amount of entropy dissipation, thus giving rise to predictions satisfying the first and second principles of thermodynamics. These techniques, however, employ regression to unveil the thermodynamic structure of the problem at the sampling points. For previously unseen situations, they rely on interpolation on the matrix manifold describing the system.

The aim of this chapter is the development of a new structure-preserving neural network architecture capable of predicting the time evolution of a system based on experimental observations on the system, with no prior knowledge of its governing equations, to be valid for both conservative and dissipative systems. The key idea is to merge the proven computational power of neural networks in highly nonlinear physics with thermodynamic consistent data-driven algorithms. The resulting methodology, as will be seen, is a powerful neural network architecture, conceptually very simple—based on standard feedforward methodologies—that exploits the right thermodynamic structure of the system as unveiled from experimental data, and that produces interpretable results [189].

3.2 Problem Statement

Weinan E seems to be the first author in interpreting the process of learning physical systems as the solution of a dynamical system [48], the so-called *dynamical systems equivalence* of machine learning. Consider a system whose governing variables will be hereafter denoted by $\boldsymbol{z} \in \mathcal{M} \subseteq \mathbb{R}^n$, with $\mathcal{M}$ the state space of these variables, which is assumed to have the structure of a differentiable manifold in $\mathbb{R}^n$. The problem of learning a given physical phenomenon can thus be seen as the one of finding an expression for the time evolution of their governing variables $\boldsymbol{z}$,

$$\dot{\boldsymbol{z}} = \frac{d\boldsymbol{z}}{dt} = F(\boldsymbol{x}, \boldsymbol{z}, t),\ \boldsymbol{x} \in \Omega \in \mathbb{R}^D,\ t \in \mathcal{I} = (0, T],\ \boldsymbol{z}(0) = \boldsymbol{z}_0, \tag{3.1}$$

where $\boldsymbol{x}$ and t refer to the space and time coordinates within a domain with $D = 2, 3$ dimensions. $F(\boldsymbol{x}, \boldsymbol{z}, t)$ is the function that gives, after a prescribed time horizon T, the flow map $\boldsymbol{z}_0 \rightarrow \boldsymbol{z}(\boldsymbol{z}_0, T)$.

While this problem can be seen as a general supervised learning problem (we fix both $\boldsymbol{z}_0$ and $\boldsymbol{z}$), when we have additional information about the physics being represented by the sought function F, it is legitimate to try to include it in the search procedure.

The goal of this chapter is to develop a new method of solving Eq. (3.1) using state-of-the-art deep learning tools, in order to predict the time evolution of the state variables of a given system. The solution is forced to fulfill the basic thermodynamic requirements of energy conservation and entropy inequality restrictions via the GENERIC formalism, presented in the next section.

3.3 Methodology

In this section we develop the appropriate thermodynamic structure for dissipative systems. The theoretical background in Section 2.3 shows that both conservative and dissipative systems can be modelled with the use of metriplectic structures such as the GENERIC formalism, based on non-equilibrium thermodynamics. It is not only a theoretically and experimentally validated theory, but also is appropriate for dynamical systems at different scales. Once such a geometric structure is found for the system, we are in the position of fixing the framework in which our neural networks can look for the adequate prediction of the future states of the system.

3.3.1 Proposed Integration Algorithm

In order to numerically solve the GENERIC equation, we formulate the discretized version of Eq. (2.10) following previous works [74]:

$$\frac{\boldsymbol{z}_{t+\Delta t} - \boldsymbol{z}_t}{\Delta t} = \boldsymbol{L}\frac{\mathrm{D}E}{\mathrm{D}\boldsymbol{z}} + \boldsymbol{M}\frac{\mathrm{D}S}{\mathrm{D}\boldsymbol{z}}. \tag{3.2}$$

The time derivative of the original equation is discretized in time increments Δt with a forward Euler scheme, where $\boldsymbol{z}_t = \boldsymbol{z}(t)$ and $T = N_T \Delta t$ with N_T the total number of snapshots of the simulation. $\boldsymbol{L}$ and $\boldsymbol{M}$ are the discretized versions of the Poisson and friction matrices. Last, $\frac{\mathrm{D}E}{\mathrm{D}\boldsymbol{z}}$ and $\frac{\mathrm{D}S}{\mathrm{D}\boldsymbol{z}}$ represent the discrete gradients, which can be approximated in a finite element sense as:

$$\frac{\mathrm{D}E}{\mathrm{D}\boldsymbol{z}} \simeq \boldsymbol{A}\boldsymbol{z}, \quad \frac{\mathrm{D}S}{\mathrm{D}\boldsymbol{z}} \simeq \boldsymbol{B}\boldsymbol{z}, \tag{3.3}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ represent the discrete matrix form of the gradient operators.

Finally, manipulating algebraically Eq. (3.2) with Eq. (3.3) and including the degeneracy conditions, the proposed integration scheme for predicting the dynamics of a physical system is the following:

$$\boldsymbol{z}_{t+\Delta t} = \boldsymbol{z}_t + \Delta t \left(\boldsymbol{L}\boldsymbol{A}\boldsymbol{z}_t + \boldsymbol{M}\boldsymbol{B}\boldsymbol{z}_t \right),$$

subject to:

$$\boldsymbol{L}\boldsymbol{B}\boldsymbol{z}_t = \boldsymbol{0},$$
$$\boldsymbol{M}\boldsymbol{A}\boldsymbol{z}_t = \boldsymbol{0},$$

ensuring the thermodynamical consistency of the resulting model. Once the learning procedure is accomplished, the neural network is expected to integrate the system dynamics in time, given previously unseen initial conditions.

To sum up, the main objective of this work is to compute the form of the $\boldsymbol{A}(\boldsymbol{z})$ and $\boldsymbol{B}(\boldsymbol{z})$ gradient operator matrices, subject to the degeneracy conditions, in order to integrate the initial system state variables $\boldsymbol{z}_0$ over certain time steps Δt of the time interval $\mathcal{I}$. In this case, the form of matrices $\boldsymbol{L}$ and $\boldsymbol{M}$ are known in advance, given the vast literature in the field. If necessary, these terms can also be computed [74], as will be explored in Chapters 4 and 5.

3.3.2 Learning Procedure

In this work, we use a multilayer perceptron as shown in Fig. 2.2, mathematically modelled as a composition of neuron functions $f^{[l]}$ following Eq. (2.1). The desired output $\hat{\boldsymbol{y}}$ is then computed as

$$\hat{\boldsymbol{y}} = (f^{[L]} \circ f^{[L-1]} \circ \ldots \circ f^{[l]} \circ \ldots \circ f^{[2]} \circ f^{[1]})(\boldsymbol{x}), \tag{3.4}$$

from a defined input $\boldsymbol{x}$ in L total layers. The challenge is to optimize the weights and biases such that they approximate the solution of the studied problem.

The input of the neural net is the vector state of a given time step $\boldsymbol{z}_t$, and the outputs are the concatenated GENERIC matrices $\boldsymbol{A}^{\text{net}}$ and $\boldsymbol{B}^{\text{net}}$, i.e. for a system with n state variables the number of inputs and outputs are $N_{\text{in}} = n$ and $N_{\text{out}} = 2n^2$. Then, using the GENERIC integration scheme, the state vector at the next time step $\boldsymbol{z}^{\text{net}}_{t+\Delta t}$ is obtained. This method is repeated for the whole simulation time T with a total of N_T snapshots.

The complete dataset $\mathcal{D}$ is composed by N_{sim} multiparametric simulation cases of a dynamical system evolving in time. Each case $\mathcal{D}_i$ contains the labelled pair of a single-step state vector $\boldsymbol{z}_t$ and its evolution in time $\boldsymbol{z}_{t+\Delta t}$ for each node of the system

$$\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^{N_{\text{sim}}}, \quad \mathcal{D}_i = \{(\boldsymbol{z}_t, \boldsymbol{z}_{t+\Delta t})\}_{t=0}^{T},$$

where the dataset $\mathcal{D}$ is disjointly partitioned in 80% training and 20% test: $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ respectively. The training is performed in a single-snapshot supervision, which has two main advantages: (i) enables parallelization between snapshots, which decreases training time by the use of GPU acceleration, and (ii) avoids intensive memory usage due to a several snapshot recursive training.

The state variables of a general dynamical system may differ in several orders of magnitude from each other, due to their own physical nature or measurement units. Then, a pre-processing of the input data (in this case, normalization) is used here to improve the model performance and stability.

The number of hidden layers N_h depends on the complexity of the problem. Increasing the net size raises the computational power of the net to model more complex phenomena. However, it slows the training process and could lead to data overfitting, limiting its generalization and extrapolation capacity. The size of the hidden layers is chosen to be the same as the output size of the net N_{out}.

The cost function for our neural network is composed of three different terms:

- **Data loss**: The main loss condition is the agreement between the network output and the real data. It is computed as the squared error sum, computed between the predicted state vector z_{t+1}^{net} and the ground truth solution $z_{t+\Delta t}^{\text{GT}}$ for each time step,

$$\mathcal{L}^{\text{data}} = \|z_{t+\Delta t}^{\text{GT}} - z_{t+\Delta t}^{\text{net}}\|_2^2, \tag{3.5}$$

  where $\|\cdot\|_2$ denotes the L2-norm.

- **Fulfilment of the degeneracy conditions**: The cost function will also account for the degeneracy conditions in order to ensure the thermodynamic consistency of the solution, implemented as the sum of the squared elements of the degeneracy vectors for each time step,

$$\mathcal{L}^{\text{deg}} = \|\boldsymbol{L}\boldsymbol{B}^{\text{net}}\boldsymbol{z}_t^{\text{net}}\|_2^2 + \|\boldsymbol{M}\boldsymbol{A}^{\text{net}}\boldsymbol{z}_t^{\text{net}}\|_2^2. \tag{3.6}$$

  This term acts as a regularization of the loss function and, at the same time, is the responsible of ensuring thermodynamic consistency. In other words, it is the cornerstone of our method.

- **Regularization**: In order to avoid overfitting, an extra L2 regularization term $\mathcal{L}^{\text{reg}}$ is added to the loss function, defined as the sum over the squared weight parameters of the network.

$$\mathcal{L}^{\text{reg}} = \sum_{l}^{L} \sum_{i}^{n^{[l]}} \sum_{j}^{n^{[l+1]}} (w_{i,j}^{[l]})^2. \tag{3.7}$$

The total cost function, Eq. (3.8), is computed as the sum squared error (SSE) of the data loss and degeneracy residual, in addition to the regularization term, at the end of the simulation time T for each train case. The regularization loss is highly dependent on the size of the network layers and has different scaling with respect to the other terms, so it is compensated with the regularization hyperparameter λ_r (weight decay). An additional weight λ_d is added to the data loss term, which accounts for the relative scaling error with respect to the degeneracy conditions.

$$\mathcal{L} = \sum_{n=0}^{N_T} (\lambda_d \mathcal{L}_n^{\text{data}} + \mathcal{L}_n^{\text{deg}}) + \lambda_r \mathcal{L}^{\text{reg}}. \tag{3.8}$$

The usual backpropagation algorithm [204] is then used to calculate the gradient of the loss function for each net parameter (weight and bias vectors), which are updated with the gradient descent technique [226]. The process is then repeated for a maximum number of epochs N_{epoch}. The resulting training algorithm is sketched in Fig. 3.1.

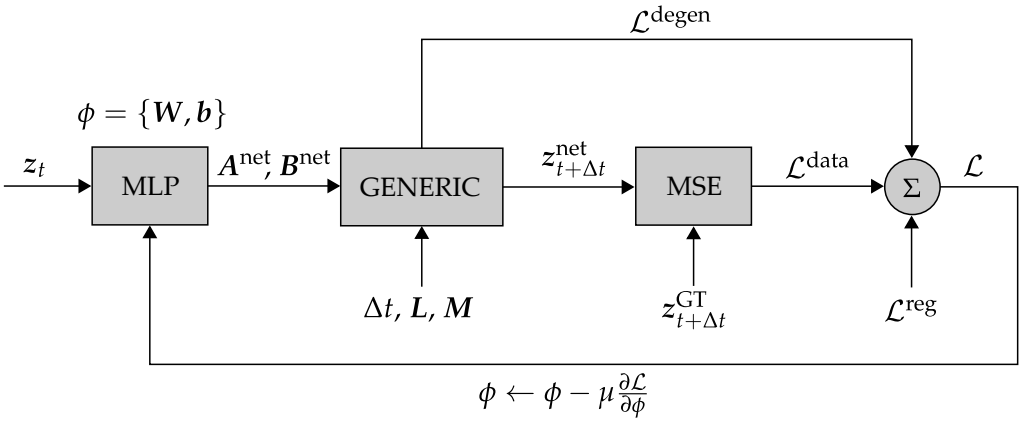


Figure 3.1: Sketch of a structure-preserving neural network training algorithm.

Algorithms 1 and 2 show a pseudocode of our proposed algorithm to both the training and test processes. The proposed method is fully implemented in PyTorch [205] and trained in an Intel Core i7-8665U CPU.

3.3.3 Evaluation Metrics

The net performance is evaluated with the mean squared error (MSE) of the state variables prediction, associated with the data loss term, Eq. (3.5), over all the time snapshots,

$$\text{MSE}^{\text{data}}(\boldsymbol{z}_i) = \frac{1}{N_T} \sum_{n=0}^{N_T} \left(z_{t,i}^{\text{GT}} - z_{t,i}^{\text{net}} \right)^2. \tag{3.9}$$

Algorithm 1: Pseudocode for the train algorithm of the SPNN.

Load train database: z^{GT} (train partition), Δt, $\boldsymbol{L}$, $\boldsymbol{M}$;
Define network architecture: N_{in}, $N_{\mathrm{out}} = 2N_{\mathrm{in}}^2$, N_h, σ_j;
Define hyperparamteres: η, λ_d, λ_r;
Initialize $w_{i,j}$, b_j;
for $epoch \leftarrow 1 : N_{\mathrm{epoch}}$ **do**
  for $train\_case \leftarrow 1 : N_{\mathrm{train}}$ **do**
    Initialize state vector: $z_0^{\mathrm{net}} \leftarrow z_0^{\mathrm{GT}}$;
    Initialize losses: $\mathcal{L}^{\mathrm{data}}, \mathcal{L}^{\mathrm{deg}} = 0$;
    for $snapshot \leftarrow 1 : N_T$ **do**
      Forward propagation: $[\boldsymbol{A}^{\mathrm{net}}, \boldsymbol{B}^{\mathrm{net}}] \leftarrow \mathrm{SPNN}(z_t^{\mathrm{GT}})$; ▷ Eq. (3.4)
      Time integration:
      $z_{t+\Delta t}^{\mathrm{net}} \leftarrow z_t^{\mathrm{net}} + \Delta t \, (\boldsymbol{L}\boldsymbol{A}^{\mathrm{net}} z_t^{\mathrm{net}} + \boldsymbol{M}\boldsymbol{B}^{\mathrm{net}} z_t^{\mathrm{net}})$; ▷ Eq. (3.2)
      Update data loss: $\mathcal{L}^{\mathrm{data}} \leftarrow \mathcal{L}^{\mathrm{data}} + \mathcal{L}_n^{\mathrm{data}}$; ▷ Eq. (3.5)
      Update degeneracy loss: $\mathcal{L}^{\mathrm{deg}} \leftarrow \mathcal{L}^{\mathrm{deg}} + \mathcal{L}_n^{\mathrm{deg}}$; ▷ Eq. (3.6)
    end for
    SSE loss function: $L \leftarrow \lambda_d \mathcal{L}^{\mathrm{data}} + \mathcal{L}^{\mathrm{deg}} + \lambda_r \mathcal{L}^{\mathrm{reg}}$ ▷ Eq. (3.7)–(3.8)
    Backward propagation;
    Optimizer step;
  end for
  Learning rate scheduler;
end for

The same procedure is applied to the degeneracy constraint, associated with the degeneracy loss term, Eq. (3.6), over all the time snapshots,

$$\mathrm{MSE}^{\mathrm{deg}}\left(z_i\right) = \frac{1}{N_T} \sum_{n=0}^{N_T} \left(\boldsymbol{L}\boldsymbol{B}^{\mathrm{net}} z_{t,i}^{\mathrm{net}} + \boldsymbol{M}\boldsymbol{A}^{\mathrm{net}} z_{t,i}^{\mathrm{net}} \right). \tag{3.10}$$

As a general error magnitude of the algorithm, the average MSE of both the train ($N = N_{\mathrm{train}}$) and test trajectories ($N = N_{\mathrm{test}}$) is also reported for both the data ($m =$ data) and degeneracy ($m =$ deg) constraints,

$$\overline{\mathrm{MSE}}^{m}\left(z\right) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{MSE}^{m}\left(z_i\right). \tag{3.11}$$

Algorithm 2: Pseudocode for the test algorithm of the SPNN.

Load test database: z^{GT} (test partition), Δt, $\boldsymbol{L}$, $\boldsymbol{M}$;
Load network parameters;
for *test_case* $\leftarrow 1 : N_{test}$ **do**
  Initialize state vector: $z_0^{\mathrm{net}} \leftarrow z_0^{\mathrm{GT}}$;
  for *snapshot* $\leftarrow 1 : N_T$ **do**
    Forward propagation: $[\boldsymbol{A}^{\mathrm{net}}, \boldsymbol{B}^{\mathrm{net}}] \leftarrow \mathrm{SPNN}(z_t^{\mathrm{net}})$; ▷ Eq. (3.4)
    Time step integration:
    $z_{t+\Delta t}^{\mathrm{net}} \leftarrow z_t^{\mathrm{net}} + \Delta t \, (\boldsymbol{L}\boldsymbol{A}_t^{\mathrm{net}} z_t^{\mathrm{net}} + \boldsymbol{M}\boldsymbol{B}_t^{\mathrm{net}} z_t^{\mathrm{net}})$; ▷ Eq. (3.2)
    Update state vector: $z_t^{\mathrm{net}} \leftarrow z_{t+\Delta t}^{\mathrm{net}}$;
  end for
  Compute $\mathrm{MSE}^{\mathrm{data}}$, $\mathrm{MSE}^{\mathrm{deg}}$; ▷ Eq. (3.9)–(3.10)
end for
Compute $\overline{\mathrm{MSE}}^{\mathrm{data}}$, $\overline{\mathrm{MSE}}^{\mathrm{deg}}$; ▷ Eq. (3.11)

3.4 Experiments

3.4.1 Double Thermoelastic Pendulum

Description

The first example is a double thermoelastic pendulum consisting of two masses m_1 and m_2 connected by two springs of variable lengths λ_1 and λ_2 and natural lengths at rest λ_1^0 and λ_2^0, as depicted in Fig. 3.2.

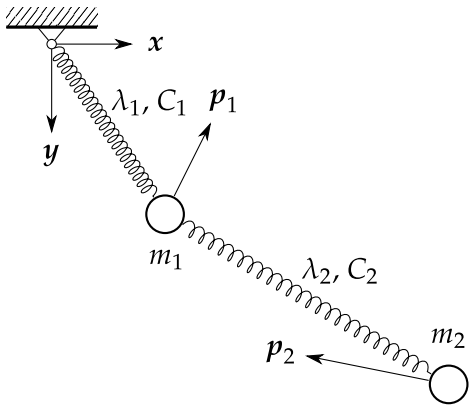


Figure 3.2: Double thermoelastic pendulum.

The set of variables describing the double pendulum is here chosen to be

$$\mathcal{S} = \{z = (\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{p}_1, \boldsymbol{p}_2, s_1, s_2) \in (\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R})\}, \tag{3.12}$$

where $\boldsymbol{q}_i$, $\boldsymbol{p}_i$ and s_i are the position, linear momentum and entropy of each mass $i = 1, 2$. Due to geometric restrictions, $\boldsymbol{q}_1 \neq \boldsymbol{0}$ and $\boldsymbol{q}_1 \neq \boldsymbol{q}_2$. The lengths of the

springs λ_1 and λ_2 are defined solely in terms of the positions as

$$\lambda_1 = \sqrt{\boldsymbol{q}_1 \cdot \boldsymbol{q}_1}, \quad \lambda_2 = \sqrt{(\boldsymbol{q}_2 - \boldsymbol{q}_1) \cdot (\boldsymbol{q}_2 - \boldsymbol{q}_1)}.$$

The total energy of the system can be expressed as the sum of the kinetic energy of the two masses K_i and the internal energy of the springs e_i for $i = 1, 2$,

$$\begin{aligned} E = E(\boldsymbol{z}) &= \sum_i K_i(\boldsymbol{z}) + \sum_i e_i(\lambda_i, s_i), \\ K_i &= \frac{1}{2m_i} |\boldsymbol{p}_i|^2. \end{aligned} \tag{3.13}$$

The total entropy of the double pendulum is the sum of the entropies of the two masses s_i,

$$S = S(\boldsymbol{z}) = s_1 + s_2. \tag{3.14}$$

This model includes thermal effects in the stretching of the springs due to the Gough-Joule effect. The absolute temperature T_i at each spring is obtained using Eq. (3.15). These temperature changes induce a heat flux between both springs, being proportional to the temperature difference and a conductivity constant $\kappa > 0$,

$$T_i = \frac{\partial e_i}{\partial s_i}. \tag{3.15}$$

In this case, there is a clear contribution of both conservative Hamiltonian mechanics (mass movement) and non-Hamiltonian dissipative effects (heat flux), resulting in a non-zero Poisson matrix ($\boldsymbol{M} \neq \boldsymbol{0}$). Thus, the GENERIC matrices associated with this physical system are known to be [74]

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{-1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{-1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 & 0 \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 1 & -1/2 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -1/2 & 1 \end{bmatrix}.$$

Database and Hyperparameters

The training database is generated with a thermodynamically consistent time stepping algorithm [222] in MATLAB. The masses of the double pendulum are set to $m_1 = 1$ kg and $m_2 = 2$ kg, joint with springs of a natural length of $\lambda_1^0 = 2$ m and

$\lambda_2^0 = 1$ m and thermal constant of $C_1 = 0.02$ J and $C_2 = 0.2$ J and conductivity constant of $\kappa = 0.5$. The simulation time of the movement is $T = 60$ s in time increments of $\Delta t = 0.3$ s ($N_T = 200$ snapshots).

The database consists of the state vector, Eq. (3.12), of 50 different trajectories with random initial conditions of position $\boldsymbol{q}_i$ and linear momentum $\boldsymbol{p}_i$ of both masses m_i ($i = 1, 2$) around a mean position and linear momentum of $\boldsymbol{q}_1 = [4.5,\ 4.5]$ m, $\boldsymbol{p}_1 = [2,\ 4.5]$ kg·m/s, and $\boldsymbol{q}_2 = [-0.5,\ 1.5]$ m, $\boldsymbol{p}_2 = [1.4,\ -0.2]$ kg·m/s respectively. Although the initial conditions of the simulations are similar, it results in a wide variety of the mass trajectories due to the chaotic behaviour of the system. This database is split randomly in 40 train trajectories and 10 test trajectories. Thus, there is a total of $8 \cdot 10^4$ training snapshots and $2 \cdot 10^4$ test snapshots.

The net input and output size is $N_{\text{in}} = 10$ and $N_{\text{out}} = 2N_{\text{in}}^2 = 200$. The state vector is normalized based on the training set statistical mean and standard deviation. The number of hidden layers is $N_h = 5$ with ReLU activation functions and linear in the last layer. It is initialized according to the Kaiming method [92] with normal distribution and the optimizer used is Adam [128] with a weight decay of $\lambda_r = 10^{-5}$ and data loss weight of $\lambda_d = 10^2$. A multistep learning rate scheduler is used, starting in $\eta = 10^{-3}$ and decaying by a factor of $\gamma = 0.1$ in epochs 600 and 1200. The training process ends when a fixed number of epochs $N_{\text{epoch}} = 1800$ is reached.

The time evolution of the data $\mathcal{L}^{\text{data}}$ and degeneracy $\mathcal{L}^{\text{deg}}$ loss terms for each training epoch are shown in Fig. 3.3.

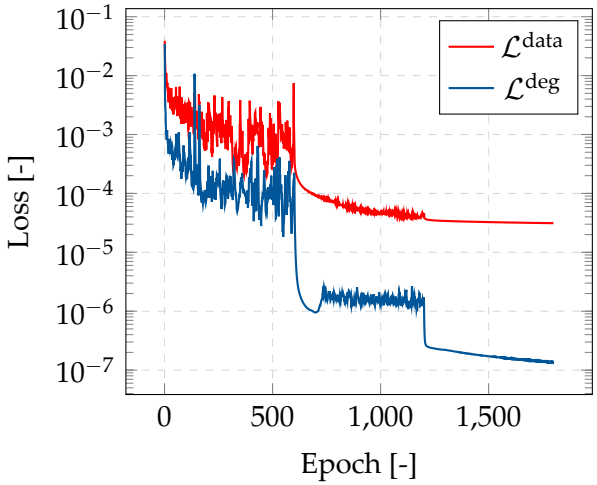


Figure 3.3: Loss evolution of data and degeneracy constraints for each epoch of the structure-preserving neural network training process of the double pendulum example.

Results

Fig. 3.4 shows the time evolution of the state variables (position, momentum and entropy) of each mass given by the solver and the neural net. Table 3.1 shows the

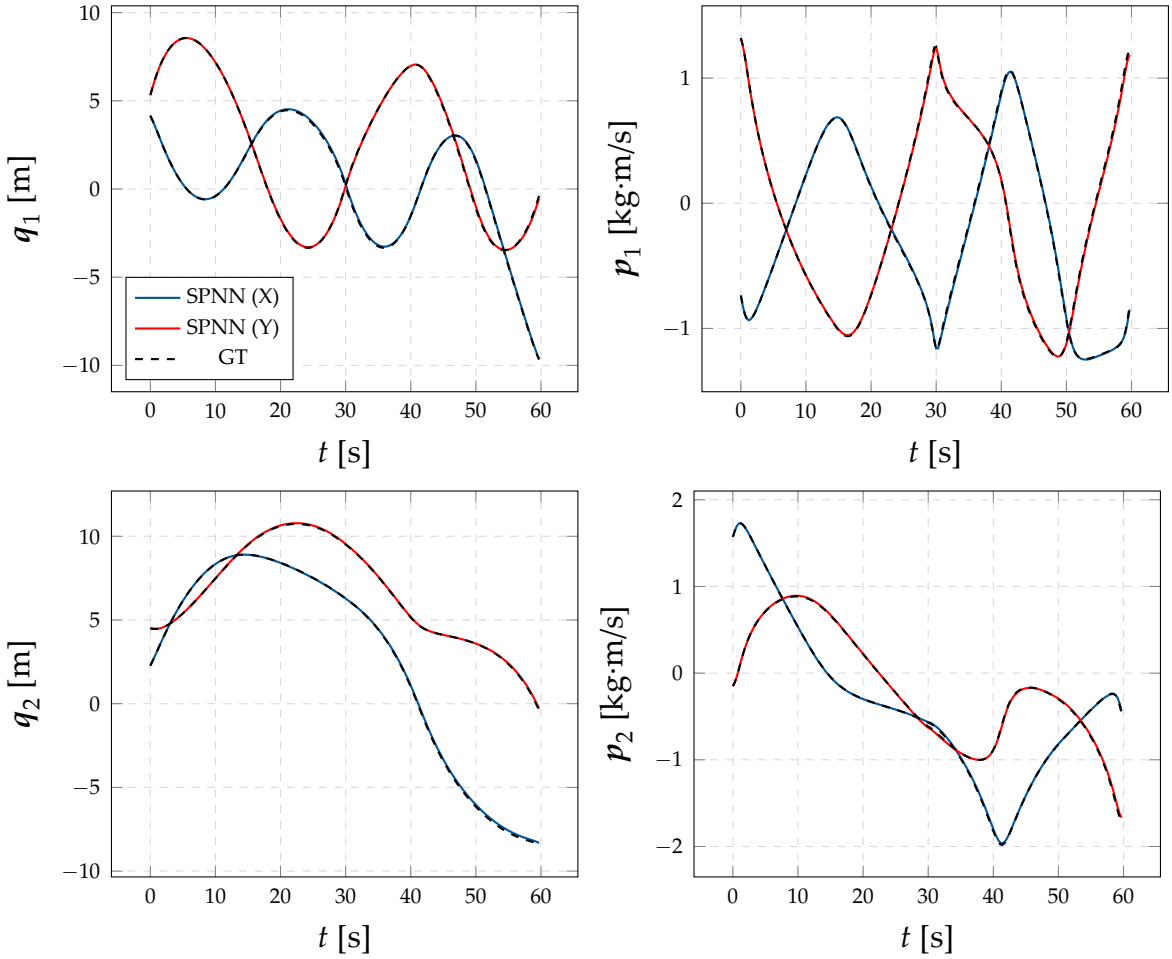


Figure 3.4: Time evolution of the state variables in a test trajectory of a double themo-elastic pendulum using a time-stepping solver (Ground Truth, GT) and the proposed GENERIC integration scheme (SPNN). Since every variable has a vectorial character, both components are depicted and labelled as X and Y, respectively.

mean squared error of the data and degeneration loss terms for all the state variables of the double pendulum. The results are computed separately as the mean over all the train and test trajectories using Eq. (3.11).

Last, Fig. 3.5 and Fig. 3.6 show the time evolution of the internal and kinetic energy and the entropy respectively for the two pendulum masses ($i = 1, 2$), computed with Eq. (3.13) and Eq. (3.14). The total energy is conserved and the total entropy satisfies the entropy inequality, fulfilling the first and second laws of thermodynamics respectively. The mean error for both train and test trajectories is reported in Table 3.2.

Table 3.1: Mean squared error of the data loss ($\overline{\text{MSE}}^{\text{data}}$) and degeneracy loss ($\overline{\text{MSE}}^{\text{deg}}$) for all the state variables of the double pendulum.

State Variables		$\overline{\text{MSE}}^{\text{data}}$		$\overline{\text{MSE}}^{\text{deg}}$	
		Train	Test	Train	Test
q_1 [m]	X	$1.95 \cdot 10^{-2}$	$3.87 \cdot 10^{-2}$	$3.56 \cdot 10^{-8}$	$4.43 \cdot 10^{-8}$
	Y	$2.72 \cdot 10^{-2}$	$8.21 \cdot 10^{-2}$	$4.74 \cdot 10^{-8}$	$5.77 \cdot 10^{-8}$
q_2 [m]	X	$2.04 \cdot 10^{-2}$	$3.65 \cdot 10^{-2}$	$9.28 \cdot 10^{-8}$	$8.55 \cdot 10^{-8}$
	Y	$2.72 \cdot 10^{-2}$	$3.73 \cdot 10^{-2}$	$3.55 \cdot 10^{-8}$	$5.01 \cdot 10^{-8}$
p_1 [kg·m/s]	X	$6.43 \cdot 10^{-4}$	$1.33 \cdot 10^{-4}$	$4.00 \cdot 10^{-8}$	$7.08 \cdot 10^{-8}$
	Y	$1.06 \cdot 10^{-3}$	$4.06 \cdot 10^{-3}$	$1.21 \cdot 10^{-7}$	$1.40 \cdot 10^{-7}$
p_2 [kg·m/s]	X	$4.88 \cdot 10^{-4}$	$9.84 \cdot 10^{-4}$	$6.00 \cdot 10^{-8}$	$4.58 \cdot 10^{-8}$
	Y	$8.47 \cdot 10^{-4}$	$1.79 \cdot 10^{-4}$	$9.76 \cdot 10^{-8}$	$1.20 \cdot 10^{-7}$
s_1 [J/K]		$1.21 \cdot 10^{-5}$	$3.51 \cdot 10^{-5}$	$1.31 \cdot 10^{-7}$	$2.06 \cdot 10^{-7}$
s_2 [J/K]		$1.22 \cdot 10^{-5}$	$3.18 \cdot 10^{-5}$	$2.40 \cdot 10^{-7}$	$2.95 \cdot 10^{-7}$

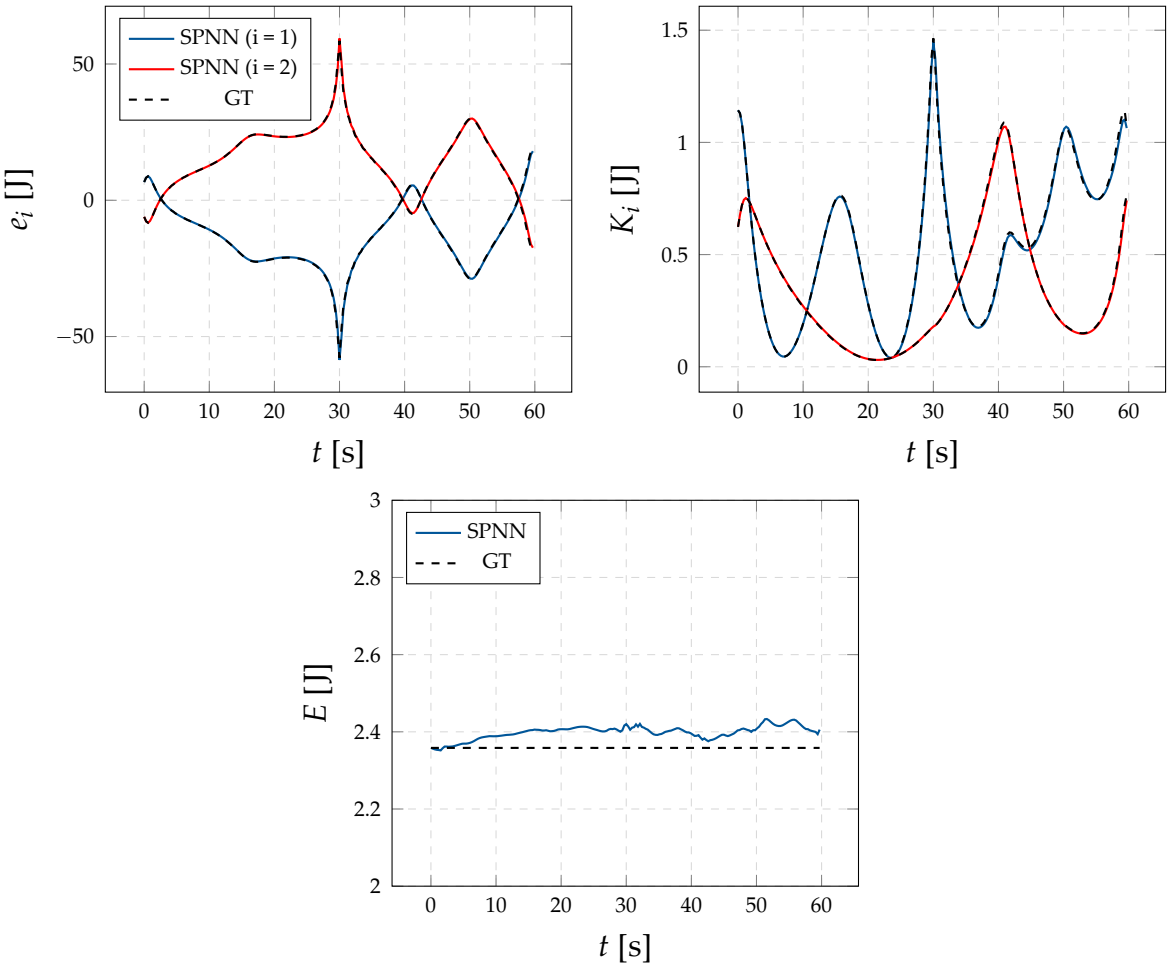


Figure 3.5: Time evolution of the energy in a test trajectory of a double themo-elastic pendulum using a time-stepping solver (Ground Truth, GT) and the proposed GENERIC integration scheme (SPNN).

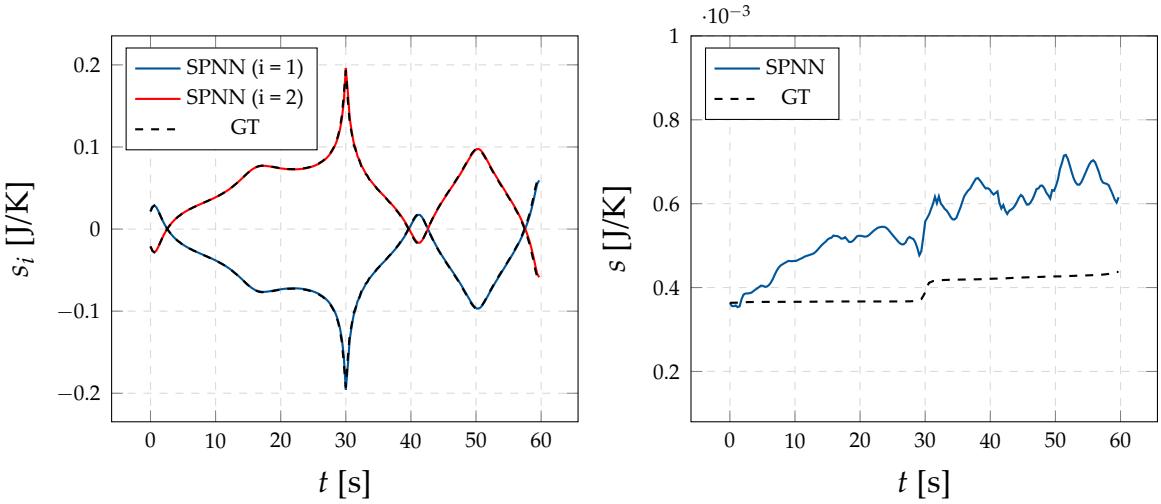


Figure 3.6: Time evolution of the entropy in a test trajectory of a double themo-elastic pendulum using a time-stepping solver (Ground Truth, GT) and the proposed GENERIC integration scheme (SPNN).

Table 3.2: Mean squared error of the energy ($\overline{\text{MSE}}$ (E)) and entropy ($\overline{\text{MSE}}$ (s)) of the double pendulum.

Variable	Train	Test
E [J]	$7.99 \cdot 10^{-3}$	$8.86 \cdot 10^{-3}$
S [J/K]	$6.52 \cdot 10^{-8}$	$6.33 \cdot 10^{-8}$

3.4.2 Couette Flow of an Oldroyd-B Fluid

Description

The second example is a shear (Couette) flow of an Oldroyd-B fluid model, see Fig. 3.7. This is a constitutive model for viscoelastic fluids, consisting of linear elastic dumbbells (representing polymer chains) immersed in a solvent.

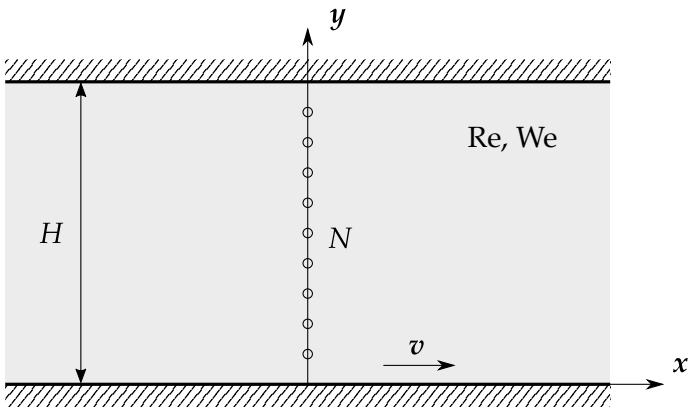


Figure 3.7: Couette flow in an Oldroyd-B fluid.

The Oldroyd-B model arises in the modelling of flows of diluted polymeric solutions. This model can be obtained both from a purely macroscopic point of view as well as from a microscopic one, by modelling polymer chains as linear dumbbells diluted in a Newtonian substrate. Alternatively, it can also be obtained by considering the deviatoric part $\boldsymbol{T}$ of the stress tensor $\boldsymbol{\sigma}$ (the so-called extra-stress tensor), to be of the form

$$\boldsymbol{T} + \lambda_1 \overset{\nabla}{\boldsymbol{T}} = \eta_0 \left(\dot{\boldsymbol{\gamma}} + \lambda_2 \overset{\nabla}{\dot{\boldsymbol{\gamma}}} \right),$$

where $\overset{\nabla}{\cdot}$ denotes the non-linear Oldroyd's upper-convected derivative [202]. Coefficients η_0, λ_1 and λ_2 are model parameters. It is standard to denote the strain rate tensor by $\dot{\boldsymbol{\gamma}} = (\boldsymbol{\nabla}^s \boldsymbol{v}) = \boldsymbol{D}$. Finally, the stress in the solvent (denoted by a subscript s) and polymer (denoted by a subscript p) are given by

$$\boldsymbol{T} = \eta_s \dot{\boldsymbol{\gamma}} + \boldsymbol{\tau},$$

so that

$$\boldsymbol{\tau} + \lambda_1 \overset{\nabla}{\boldsymbol{\tau}} = \eta_p \dot{\boldsymbol{\gamma}},$$

which is the constitutive equation for the elastic stress.

Pseudo-experimental data are obtained by the CONNFFESSIT technique [138], based on the Fokker-Plank equation [139]. This equation is solved by converting it in its

corresponding Itô stochastic differential equation,

$$dr_x = \left(\frac{\partial v}{\partial y} r_y - \frac{1}{2\mathrm{We}} r_x\right) dt + \frac{1}{\sqrt{\mathrm{We}}} dV_t,$$
$$dr_y = -\frac{1}{2\mathrm{We}} r_y dt + \frac{1}{\sqrt{\mathrm{We}}} dW_t,$$

where v is the flow velocity, $\boldsymbol{r} = [r_x,\ r_y]$, $r_x = r_x(y,t)$ the position vector and assuming a Couette flow so that $r_y = r_y(t)$ depends only on time, We stands for the Weissenberg number and V_t, W_t are two independent one-dimensional Brownian motions. This equation is solved via Monte Carlo techniques, by replacing the mathematical expectation by the empirical mean.

The model relies on the microscopic description of the state of the dumbbells. Thus, it is particularly useful to base the microscopic description on the evolution of the conformation tensor $\boldsymbol{c} = \langle \boldsymbol{rr} \rangle$, this is, the second moment of the dumbbell end-to-end distance distribution function. This tensor is in general not experimentally measurable and plays the role of an internal variable. The expected xy stress component tensor will be given by

$$\tau = \frac{\epsilon}{\mathrm{We}} \frac{1}{K} \sum_{k=1}^{K} r_x r_y,$$

where K is the number of simulated dumbbells and $\epsilon = \frac{\nu_p}{\nu_p}$ is the ratio of the polymer to solvent viscosities.

The state variables selected for this problem are the position of the fluid on each node of the mesh, see Fig. 3.7, its velocity v in the x direction, internal energy e and the conformation tensor shear component τ,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{v}, e, \tau) \in (\mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \times \mathbb{R})\}. \tag{3.16}$$

The GENERIC matrices associated with each node of this physical system are the following

$$\boldsymbol{L} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

In order to simulate a measurement of real captured data, Gaussian noise is added to the state vector, computed as a random variable following a normal distribution with zero mean and standard deviation proportional to the standard deviation of

the database σ_z and noise level ν,

$$z^{\mathrm{GT}}_{\mathrm{noise}} = z^{\mathrm{GT}} + \nu \cdot \sigma_z \cdot \mathcal{N}(0,1)$$

The results of both the noise-free and the noisy database are compared with two different network architectures:

- Unconstrained network: This architecture is the same as the proposed network but removing the degeneracy conditions of the energy and entropy, Eq. (3.6), in the loss function. These conditions ensure the thermodynamic consistency of the resulting integrator, so not including them affects negatively in the accuracy of the results, as will be seen.
- Black-Box network: In this case, no GENERIC architecture is imposed, acting as a black-box integrator trained to directly predict the state vector time evolution $z_{t+\Delta t}$ from the previous time step z_t. This naive approach is shown to be inappropriate, as no physical restrictions are given to the model.

Database and Hyperparameters

The training database for this Oldroyd-B model is generated in MATLAB with a multiscale approach [139] in the dimensionless form. The fluid is discretized in the vertical direction with $N = 100$ elements (101 nodes) in a total height of $H = 1$. A total of 10^4 dumbells were considered at each nodal location in the model. The lid velocity is set to $V = 1$, the viscoelastic Weissenberg number We $= 1$ and Reynolds number of Re $= 0.1$. The simulation time of the movement is $T = 1$ in time increments of $\Delta t = 6.7 \cdot 10^{-3}$ ($N_T = 150$ snapshots).

The database consisted of the state vector, Eq. (3.16), of the 100 nodes trajectories (excluding the node at $h = H$, for which a no-slip condition $v = 0$ has been imposed). This database is split in 80 train trajectories and 20 test trajectories.

The net input and output size is $N_{\mathrm{in}} = 5$ and $N_{\mathrm{out}} = 2N^2_{\mathrm{in}} = 50$. The number of hidden layers is $N_h = 5$ with ReLU activation functions and linear in the last layer. It is initialized according to the Kaiming method [92], with normal distribution and the optimizer used is Adam [128], with a weight decay of $\lambda_r = 10^{-5}$ and data loss weight of $\lambda_d = 10^3$. A multistep learning rate scheduler is used, starting in $\eta = 10^{-3}$ and decaying by a factor of $\gamma = 0.1$ in epochs 500 and 1000. The training process ends when a fixed number of epochs $N_{\mathrm{epoch}} = 1500$ is reached. The same parameters are considered also for the noisy database network ($\nu = 1\%$) and the unconstrained network.

The black-box network training parameters are analogous to the structure-preserving network, except for the output size $N_{\mathrm{out}} = N_{\mathrm{in}} = 5$. Several network architectures were tested, and the lowest error is achieved with $N_h = 5$ hidden layers and 25 neurons each layer.

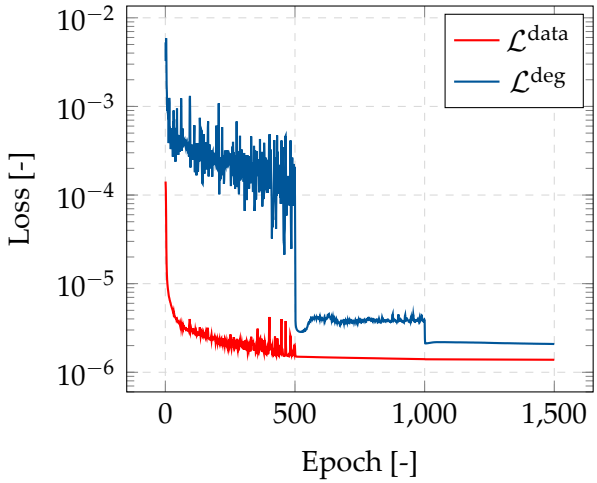


Figure 3.8: Loss evolution of data and degeneracy constraints for each epoch of the neural network training process of the Couette flow example.

The time evolution of the data $\mathcal{L}^{\text{data}}$ and degeneracy $\mathcal{L}^{\text{deg}}$ loss terms for each training epoch are shown in Fig. 3.8.

Results

Fig. 3.9 shows the time evolution of the state variables (position q, velocity v, internal energy e and conformation tensor shear component τ) given by the solver and the neural net. There is a good agreement between both plots. Moreover, the proposed scheme is able to predict the time evolution of the flow for several snapshots beyond the training simulation time $T = 1$, as shown in the same figure.

Table 3.3 shows the mean squared error of the data and degeneration loss terms for all the state variables of the Couette flow of an Oldroyd-B fluid. The results are computed separately as the mean over all the train and test trajectories using Eq. (3.11).

Table 3.3: Mean squared error of the data loss ($\overline{\text{MSE}}^{\text{data}}$) and degeneracy loss ($\overline{\text{MSE}}^{\text{deg}}$) for all the state variables of the Couette flow.

State Variables		$\overline{\text{MSE}}^{\text{data}}$		$\overline{\text{MSE}}^{\text{deg}}$	
		Train	Test	Train	Test
q [-]	X	$5.40 \cdot 10^{-6}$	$6.29 \cdot 10^{-6}$	$1.72 \cdot 10^{-7}$	$1.96 \cdot 10^{-7}$
	Y	0.00	0.00	0.00	0.00
v [-]		$3.23 \cdot 10^{-5}$	$4.75 \cdot 10^{-5}$	$1.19 \cdot 10^{-6}$	$1.48 \cdot 10^{-6}$
e [-]		$7.85 \cdot 10^{-6}$	$6.60 \cdot 10^{-6}$	$7.06 \cdot 10^{-7}$	$9.11 \cdot 10^{-7}$
τ [-]		$2.36 \cdot 10^{-5}$	$1.26 \cdot 10^{-5}$	$1.07 \cdot 10^{-6}$	$1.31 \cdot 10^{-6}$

Fig. 3.10 shows a box plot of the data error (MSE^{data}) for the train and test sets in the four studied architectures. The results of the structure-preserving neural network

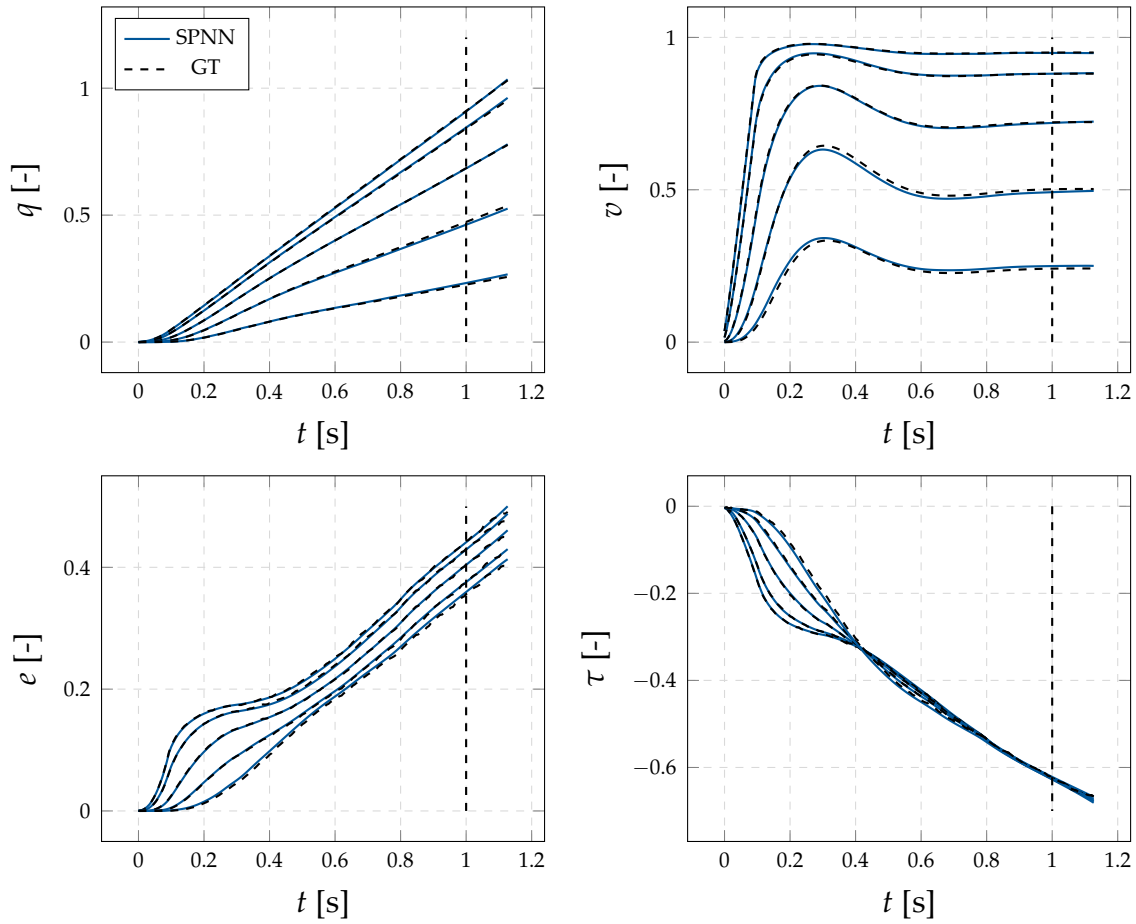


Figure 3.9: Time evolution of the state variables in five test nodes of a Couette flow using a solver (Ground Truth, GT) and the proposed GENERIC integration scheme (Net). The dotted vertical line represent the simulation time $T = 1$ of the training dataset.

outperform the other two approaches even with noisy training data. The error of the unconstrained neural network is greater than one order of magnitude than our approach, proving the importance of the degeneracy conditions in the GENERIC formulation. Last, the naive black-box approach shows the worst performance of the four networks, as no physical restriction is considered.

With respect to our previous work [74], that employed a piece-wise linear regression approach, these examples show similar levels of accuracy, but a much greater level of robustness. For instance, this same example was included in the mentioned reference. However, in that case, the problem had to be solved with the help of a reduced order model with only six degrees of freedom, due to the computational burden of the approach. In our former approach, the GENERIC structure was identified by piece-wise linear regression for each of the few global *modes* of the approximation. So to speak, in that case, we *learnt* the characteristics of the flow. Here, on the contrary, the net is able to find an approximation for any velocity value at the 101 nodes of the mesh—say, fluid particles—without any difficulty. In this case, we are *learning* the behavior of fluid particles.

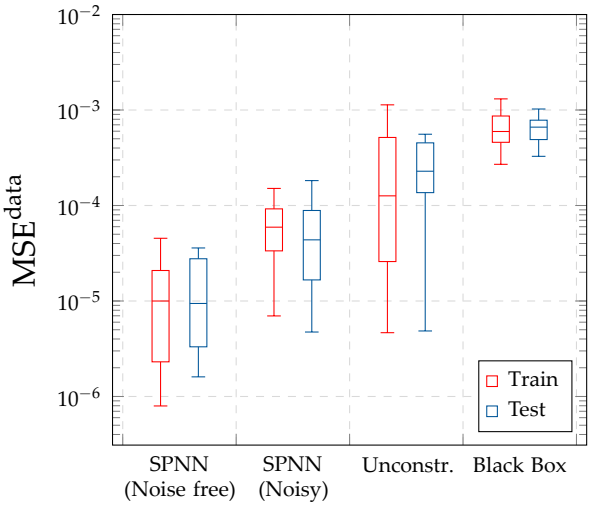


Figure 3.10: Box plots for the data integration mean squared error ($\mathrm{MSE}^{\mathrm{data}}$) of the Couette flow in both train and test cases.

3.5 Conclusions

In this chapter we have presented a new methodology to ensure thermodynamic consistency in the deep learning of physical phenomena. In contrast to existing methods, this methodology does not need to know in advance any information related to balance equations or the precise form of the PDE governing the phenomena at hand. The method is constructed on top of the right thermodynamic principles that ensure the fulfillment of the energy dissipation and entropy production. It is valid, therefore, for conservative as well as dissipative systems, thus overcoming previous approaches in the field.

When compared with other previous works in the field [74], the present methodology showed to be more robust, allowing us to find approximations for systems with orders of magnitude more degrees of freedom. This new approach is also less computationally demanding. For the double pendulum case, the snapshot optimization of the GENERIC matrices proposed in [74] has a measured performance of 10 min per trajectory, which add up to 400 minutes considering the 40 studied trajectories, whereas our new neural-network approach trains in only 73.18 minutes. The computational time of the other examples is shown in Table 3.4.

The reported results show good agreement between the network output and the synthetic ground truth solution, even with moderate noisy data. We have also shown the importance of including the degeneracy conditions of the GENERIC formulation to the neural network constraints, as it ensures the thermodynamical consistency of the integrator. The structure-preserving neural network outperforms other naive black-box approaches, since the physical constraints act as an inductive bias, facilitating the learning process.

Table 3.4: Computation training time of the proposed algorithm for the two reported examples in the noise-free networks.

Example	Epoch Time	Total Time
Double Pendulum	2.44 s/epoch	73.18 min
Couette Flow	1.22 s/epoch	30.53 min

However, this method has several limitations. The discrete gradients of the state vector in the GENERIC formulation are here approximated with two gradient operators. Furthermore, the Poisson and friction matrices are considered to be known in advance. This is not the general case, because the dynamical data might have completely unknown governing equations or the integration scheme is applied to latent vectors in a reduced order model. These limitations are addressed in the next chapters.

Thermodynamics-Informed Graph Neural Networks

4

In this chapter we extend the formulation in Chapter 3 using both geometric and thermodynamic inductive biases to improve accuracy and generalization of the resulting integration scheme. The geometric prior is achieved with graph neural networks, which induce a non-Euclidean geometrical prior with permutation invariant node and edge update functions. Several examples are provided in both Eulerian and Lagrangian descriptions in the context of fluid and solid mechanics respectively, achieving relative mean errors of less than 3% in all the tested examples. Two ablation studies are also performed based on recent works in both physics-informed and geometric deep learning. The content of this chapter is included in the following publication [98]:

Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
Thermodynamics-informed graph neural networks
IEEE Transactions on Artificial Intelligence, DOI: 10.1109/TAI.2022.3179681 (2022)

4.1 Introduction

We have already seen in Chapter 3 that it is possible to learn the correct thermodynamical structure of simple systems in order to predict their evolution in time. However, real world physical systems might require the use of unstructured grids in order to capture the complex geometric details of the domain. So it is not only important to take into account the mathematical structure of the problem, but also the geometrical domain in which it is formulated.

In Section 2.1 we already saw how the first simple deep learning architectures, such as the multilayer perceptron and the vanilla autoencoder, were modified with additional inductive biases that account for the peculiarities of the data structure of

each problem. These kind of architectures are studied in the geometric deep learning field, which treats all sort of data structures in terms of graph and group theory, as explained in Section 2.2.

The present chapter aims to extend the thermodynamical biases developed in Chapter 3 with additional geometrical biases based on the principles of geometric deep learning, in order to learn a physical simulator of complex systems in the context of fluid and solid mechanics.

4.2 Methodology

In this chapter, we again guarantee the physical meaning of the solution by enforcing the GENERIC structure of the system. We make use of this constraint to construct a thermodynamically-sound integrator, acting as the first inductive bias of our approach called the metriplectic bias. This integration is performed using a forward Euler scheme with time increment Δt and the GENERIC formalism in Eq. (2.10), resulting in the following expression

$$z_{t+\Delta t} = z_t + \Delta t \left(\boldsymbol{L} \frac{\partial E}{\partial z_t} + \boldsymbol{M} \frac{\partial S}{\partial z_t} \right). \tag{4.1}$$

Unlike in Chapter 3, here we consider that no prior information about the $\boldsymbol{L}$ and $\boldsymbol{M}$ operators is known. We also do not approximate the discrete gradients, but we learn the energy E and entropy S potentials of the particles in the domain. We propose the use of graph-based deep learning, which exploits the geometrical structure of that specific domain.

4.2.1 Geometric Structure: Graph Neural Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{u})$ be a directed *graph*, where $\mathcal{V} = \{1, ..., n\}$ is a set of $|\mathcal{V}| = n$ vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of $|\mathcal{E}| = e$ edges and $\boldsymbol{u} \in \mathbb{R}^{F_g}$ is the global feature vector. Each vertex and edge in the graph is associated with a node and a pairwise interaction between nodes respectively in a discretized physical system. The global feature vector defines the properties shared by all the nodes in the graph, such as gravity or elastic properties. For each vertex $i \in \mathcal{V}$ we associate a feature vector $\boldsymbol{v}_i \in \mathbb{R}^{F_v}$, which represents the physical properties of each individual node. Similarly, for each edge $(i, j) \in \mathcal{E}$ we associate an edge feature vector $\boldsymbol{e}_{ij} \in \mathbb{R}^{F_e}$.

In practice, the positional state variables of the system ($\boldsymbol{q}_i$) are assigned to the edge feature vector $\boldsymbol{e}_{ij}$ so the edge features represent relative distances ($\boldsymbol{q}_{ij} = \boldsymbol{q}_i - \boldsymbol{q}_j$) between nodes, giving a distance-based attentional flavour to the graph network [181, 259, 279] and translational invariance [268, 237]. The rest of the state variables are assigned to the node feature vector $\boldsymbol{v}_i$. The external interactions, such as forces

applied to the system, are included in an external load vector $\boldsymbol{f}_i$. A simplified scheme of the graph codification of a physical system is depicted in Fig. 4.1.

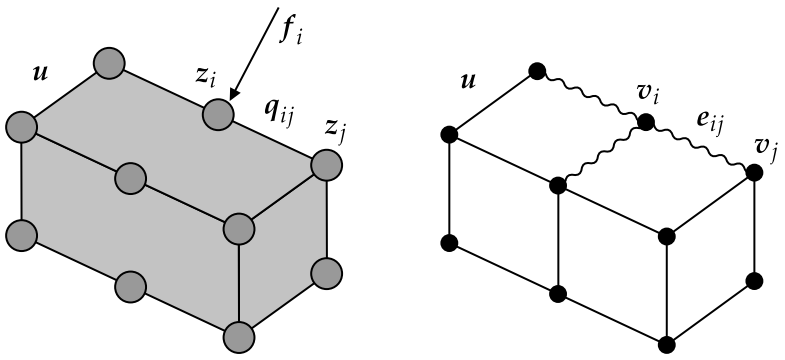


Figure 4.1: Left: Physical system domain discretized in a mesh with node state variables $\boldsymbol{z}_i$, relative nodal distances $\boldsymbol{q}_{ij}$, external interactions $\boldsymbol{f}_i$ and global properties $\boldsymbol{u}$. Right: Graph representation of the same system, with node and edge attributes: $\boldsymbol{v}_i$ and $\boldsymbol{e}_{ij}$.

These features are fed into an *encode-process-decode* scheme [14], which consists on several multilayer perceptrons (MLPs) shared between all the nodes and edges of the graph. The algorithm consists of five steps (Fig. 4.2):

Encoding

We use two MLPs (ε_v, ε_e) to transform the vertex and edge initial feature vectors into higher-dimensional embeddings $\boldsymbol{x}_i \in \mathbb{R}^{F_h}$ and $\boldsymbol{x}_{ij} \in \mathbb{R}^{F_h}$ respectively,

$$\begin{aligned} \varepsilon_e : \mathbb{R}^{F_e} &\longrightarrow \mathbb{R}^{F_h} & \qquad \varepsilon_v : \mathbb{R}^{F_v} &\longrightarrow \mathbb{R}^{F_h} \\ \boldsymbol{e}_{ij} &\longmapsto \boldsymbol{x}_{ij}, & \qquad \boldsymbol{v}_i &\longmapsto \boldsymbol{x}_i. \end{aligned}$$

Processing

The processor is the core task of the algorithm, as it shares the nodal information between vertices via message passing and modifies the hidden vectors in order to extract the desired output of the system. First, a MLP (π_e) computes the updated edge features $\boldsymbol{x}'_{ij}$ for each graph edge, based on the current edge features, global features, and sending and recieving node,

$$\begin{aligned} \pi_e : \mathbb{R}^{3F_h+F_g} &\longrightarrow \mathbb{R}^{F_h} \\ (\boldsymbol{x}_{ij}, \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{u}) &\longmapsto \boldsymbol{x}'_{ij}. \end{aligned}$$

Then, for each node the messages are pooled with a permutation invariant function ϕ based on the neighborhood $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$ of the node i. Last, the node embeddings are updated with a second MLP (π_v) using the current node features,

the pooled messages, the external load vector and the global features,

$$\pi_v : \mathbb{R}^{2F_h+F_f+F_g} \longrightarrow \mathbb{R}^{F_h}$$
$$(\boldsymbol{x}_i, \phi(\boldsymbol{x}'_{ij}), \boldsymbol{f}_i, \boldsymbol{u}) \longmapsto \boldsymbol{x}'_i,$$

where $\boldsymbol{x}'_i$ and $\boldsymbol{x}'_{ij}$ are the updated nodal and edge latent vectors.

The processing step is equivalent to the message passing [69] of 1-step adjacent nodes. In order to get the influence of further graph nodes, the process can be recurrently repeated with both shared or unshared parameters in M processing blocks and optionally using residual connections [91]. In this approach, we use both unshared parameters and residual connections to each message passing block and sum as aggregation function ϕ. Note that the computed messages $\phi(\boldsymbol{x}'_{ij})$ represent a hidden embedding of the intermolecular interactions of the system (internal messages) whereas the vector $\boldsymbol{f}_i$ accounts for the external interactions (external messages).

Decoding

The last block extracts the relevant physical output information $\boldsymbol{y}_i \in \mathbb{R}^{F_y}$ of the system from the node latent feature vector, implemented with a MLP (δ_v). In this work, we predict for each particle the GENERIC energy E and entropy S potentials and the flattened operators $\boldsymbol{l}$ and $\boldsymbol{m}$:

$$\delta_v : \mathbb{R}^{F_h} \longrightarrow \mathbb{R}^{F_y}$$
$$\boldsymbol{x}'_i \longmapsto \boldsymbol{y}_i = (\boldsymbol{l}, \boldsymbol{m}, E, S).$$

Reparametrization

A last processing step is needed to get the GENERIC parameters before integrating the state variables. Both operators in matrix form $\boldsymbol{L}$ and $\boldsymbol{M}$ are constructed using the flattened output of the graph neural network $\boldsymbol{l}$ and $\boldsymbol{m}$ respectively, reshaped in lower-triangular matrices. The skew-symmetric and positive semi-definite conditions are imposed by construction using the following parametrization:

$$\boldsymbol{L} = \boldsymbol{l} - \boldsymbol{l}^\top, \qquad \boldsymbol{M} = \boldsymbol{m}\boldsymbol{m}^\top.$$

Both E and S are directly predicted for every node. Then, these potentials can be differentiated with respect to the network input in order to get the gradients $\frac{\partial E}{\partial \boldsymbol{z}}$ and $\frac{\partial S}{\partial \boldsymbol{z}}$ needed for the GENERIC integrator. These gradients are easily obtained using automatic differentiation [204], and ensures the integrability of the energy and entropy gradients [247].

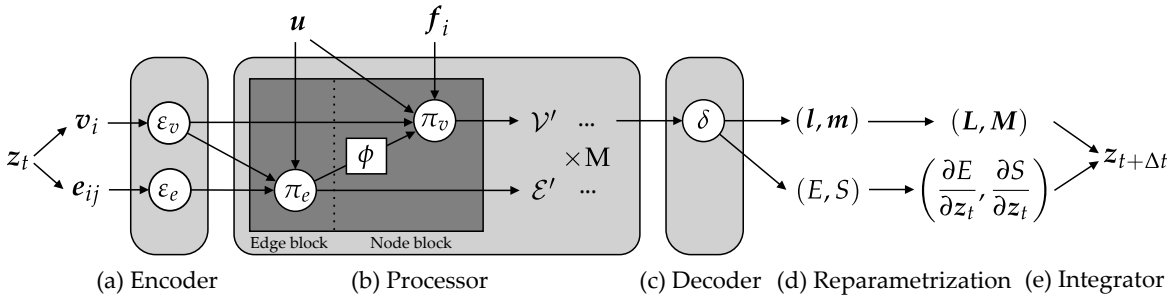


Figure 4.2: Algorithm block scheme used to predict a single-step state variable change in time. (a) The encoder transforms the node and edge features to a learnt embedding. (b) The processor shares the nodal information through the graph via M message passing modules. (c) The decoder extracts the GENERIC flattened operators and potentials from the processed node embeddings. (d) The reparametrization step builds the symmetries of the $\boldsymbol{L}$ and $\boldsymbol{M}$ operators and computes the potential gradients with respect to the network input. (e) The integrator predicts the next time step state variables based on the GENERIC formulation. The whole process is repeated iteratively to get the dynamical rollout of the physical system.

Considering the dimensions of the lower triangular matrices and the scalar value of both potentials, the output dimension of the decoder network is

$$F_y = \frac{n(n+1)}{2} + \frac{n(n-1)}{2} + 1 + 1,$$

where n represents the dimension of the state variables $\boldsymbol{z}$.

Integration

The single-step integration of the state variables of the system $\boldsymbol{z}_t \rightarrow \boldsymbol{z}_{t+\Delta t}$ is then performed using Eq. (4.1).

4.2.2 Learning Procedure

In this case, the complete dataset $\mathcal{D}$ is again composed by N_{sim} multiparametric simulation cases of a dynamical system evolving in time, in a one-step supervision similar to the previous chapter. The partition in this case is 80% training, 10% test and 10% validation sets: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$ respectively. The loss function is divided in two terms:

- **Data loss**: This term accounts for the correct prediction of the state vector time evolution using the GENERIC integrator. It is defined as the MSE along the graph nodes and state variables between the predicted and the ground-truth time derivative of the state vector in a given snapshot,

$$\mathcal{L}^{\text{data}} = \left\| \frac{d\boldsymbol{z}^{\text{GT}}}{dt} - \frac{d\boldsymbol{z}^{\text{net}}}{dt} \right\|_2^2.$$

The choice of the time derivative instead of the state vector itself is to regularize the global loss function to a uniform order of magnitude with respect to the degeneracy terms, as shown in Eq. (4.1).

- **Degeneracy loss**: This condition is added to the optimization in order to force the degeneracy conditions of the Poisson and dissipative operators, which ensure thermodynamical consistency of the integrator. It is defined as the MSE along the graph nodes and state variables of two residual terms corresponding to the energy and entropy degeneracy conditions,

$$\mathcal{L}^{\text{deg}} = \left\| \boldsymbol{L} \frac{\partial S}{\partial \boldsymbol{z}_t} \right\|_2^2 + \left\| \boldsymbol{M} \frac{\partial E}{\partial \boldsymbol{z}_t} \right\|_2^2.$$

  Alternative approaches were later developed in the literature to impose these degeneracy restrictions, such as a specific tensor parametrization of the brackets [143] or forcing orthogonality using additional skew-symmetric matrices [282]. However, we decide to include it as a soft constraint in order to allow more flexibility in the learning process and improve convergence while maintaining the degeneracy conditions up to an admissible error.

The global loss term is a weighted mean of the two terms over the shuffled N_{batch} batched snapshots,

$$\mathcal{L} = \frac{1}{N_{\text{batch}}} \sum_{n=0}^{N_{\text{batch}}} (\lambda_d \mathcal{L}_n^{\text{data}} + \mathcal{L}_n^{\text{deg}}).$$

As the energy and entropy are supervised only by their gradients, we remark that (i) they are learnt up to an integration constant value and (ii) the activation functions must have a sufficient degree of continuity. To meet this second requirement, one must select activations with non-zero second derivative in order to have a correct backpropagation of the weights and biases. Thus, linear or rectified units (ReLU, Leaky ReLU, RReLU) are not appropriate for this task. It is well known [105] that logistic functions such as sigmoid and hyperbolic tangent are universal approximators of any derivative arbitrarily well, but are not optimal for very deep neural network architectures, as they suffer from several problems such as vanishing gradients. Then, the correct activation functions suitable for learning gradients are the ones which combine both non-zero second derivatives and ReLU-type non-linearities, such as Softplus, Swish [215] or Mish [177]. In the present work we use the Swish activation function.

The inputs and outputs of the networks are standardized using the training dataset statistics. Gaussian noise is also added to the inputs during training in order to model the accumulation of error during the time integration [207], which is not contemplated in a single-snapshot training, with the variance of the noise σ^2_{noise} as a tunable hyperparameter and zero mean value. All the cases are optimized using

Adam [128] and a multistep learning rate scheduler. The code is fully implemented in Pytorch.

4.2.3 Evaluation Metrics

Two ablation studies are performed to evaluate the method presented in this work, from now thermodynamics-informed graph neural networks (TIGNN). The first case is performed using only graph neural networks (from now, GNN) with similar architecture and learning procedure used in prior works [231, 207] and no metriplectic integrator. In the second case, we impose the metriplectic structure developed in Chapter 3 (from now, SPNN), using standard MLPs with no graph computations. Both alternative methods are tuned for equal parameter count in order to get a fair comparison of the results.

All the results are computed with the integration scheme in Eq. (4.1) iteratively from the initial conditions to the prescribed time horizon T, denoted as rollout. The rollout prediction error is quantified by the relative L2 error, computed with Eq. (4.2) for each snapshot and simulation case,

$$\varepsilon = \frac{\|z^{\mathrm{GT}} - z^{\mathrm{net}}\|_2}{\|z^{\mathrm{GT}}\|_2}. \tag{4.2}$$

The results are represented in Fig. 4.8, 4.9 and 4.10 showing the rollout statistics for all the snapshots divided in train and test simulations, state variables and method used (TIGNN, GNN or SPNN).

4.3 Experiments

4.3.1 Couette Flow of an Oldroyd-B Fluid

Description

The first example is the same Couette flow as in Chapter 3. For the complete description of the system, refer to Section 3.4.2. The state variables chosen are again the position of the fluid on each node of the mesh $\boldsymbol{q}$, its velocity v in the x direction, internal energy e and the conformation tensor shear component τ, as in Eq. (5.13).

The edge feature vector contains the relative position of the nodes whereas the rest of the state variables are part of the node feature vector. An additional one-hot vector $\boldsymbol{n}$ is added to the node features in order to represent the boundary and fluid nodes. The global feature vector $\boldsymbol{u}$ represent the Weissenberg and Reynolds numbers of each simulation, resulting in the following feature vectors:

$$\boldsymbol{e}_{ij} = (\boldsymbol{q}_i - \boldsymbol{q}_j, \|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2), \quad \boldsymbol{v}_i = (v, e, \tau, \boldsymbol{n}), \quad \boldsymbol{u} = (\mathrm{Re}, \mathrm{We}). \tag{4.3}$$

Database and Hyperparameters

The training database for the Couette flow is generated with the CONNFFESSIT technique [138], based on the Fokker-Plank equation [139], using a Monte Carlo algorithm. The fluid is discretized in the vertical direction with $N_e = 100$ elements and $N = 101$ nodes in a total height of $H = 1$. A total of 10,000 dumbells are considered at each nodal location in the model. The lid velocity is set to $V = 1$, with variable Weissenberg We $\in [1, 2]$ and Reynolds number Re $\in [0.1, 1]$, summing a total of $N_{\text{sim}} = 100$ cases. The simulation is discretized in $N_T = 150$ time increments of $\Delta t = 6.7 \cdot 10^{-3}$.

Following Eq. (4.3), the dimensions of the graph feature vectors are $F_e = 3$, $F_v = 5$ and $F_g = 2$. The hidden dimension of the node and edge latent vectors is $F_h = 10$. The learning rate is set to $l_r = 10^{-3}$ with decreasing order of magnitude on epochs 2000 and 4000, and a total number of $N_{\text{epoch}} = 6000$. The training noise variance is set to $\sigma^2_{\text{noise}} = 10^{-2}$.

Results

The rollout results for the Couette flow are presented in Fig. 4.8. A substantial improvement is shown in the present approach over the two other methods, which remain in a similar performance. Note that the skewed distributions towards higher errors on each box is due to the error accumulation on snapshots further in time from the starting conditions, where errors are lower. Fig. 4.7 (left) shows that the degeneracy conditions imposed by our method ensure the thermodynamical consistency of the learnt energy and entropy potentials.

4.3.2 Viscoelastic Bending Beam

Description

The next example is a prismatic viscoelastic cantilever beam subjected to a bending force, as depicted in Fig. 4.3. The state variables for the viscoelastic beam on each node are the position $\boldsymbol{q}$, velocity $\boldsymbol{v}$ and stress tensor $\boldsymbol{\sigma}$,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\sigma}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^6\}.$$

The material is characterized by a single-term polynomial strain energy potential, described by the following equation

$$U = C_{10}(\bar{I}_1 - 3) + C_{01}(\bar{I}_2 - 3) + \frac{1}{D_1}(J_{el} - 1)^2$$

where U is the strain energy potential, J_{el} is the elastic volume ratio, $\bar{I}_1$ and $\bar{I}_2$ are the two invariants of the left Cauchy-Green deformation tensor, C_{10} and C_{01} are shear

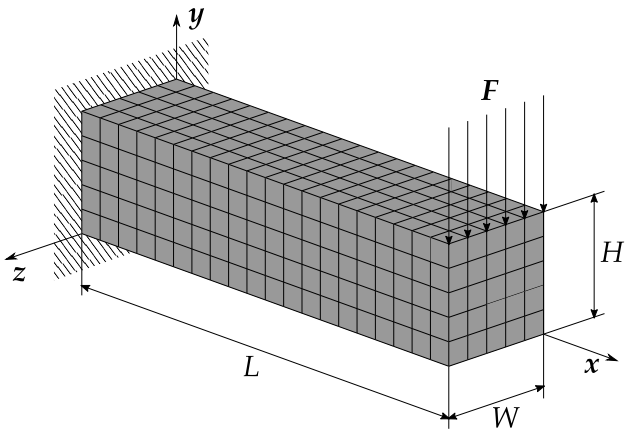


Figure 4.3: Viscoelastic beam problem with a load case. The load position and direction are modified on each simulation, obtaining different stress fields.

material constants and D_1 is the material compressibility parameter. The viscoelastic component is described by a two-term Prony series of the dimensionless shear relaxation modulus,

$$g_R(t) = 1 - \bar{g}_1(1 - e^{\frac{-t}{\tau_1}}) - \bar{g}_2(1 - e^{\frac{-t}{\tau_2}}),$$

with relaxation coefficients of $\bar{g}_1$ and $\bar{g}_2$, and relaxation times of τ_1 and τ_2.

The relative deformed position is included into the edge feature vector whereas the rest of the variables are part of the node feature vector. An additional one-hot vector $\boldsymbol{n}$ is added to the node features in order to represent the encastre and beam nodes. The external load vector $\boldsymbol{F}$ is included in the node processor MLP as an external interaction. No global feature vector is needed in this case, resulting in the following feature vectors:

$$\boldsymbol{e}_{ij} = (\boldsymbol{q}_i - \boldsymbol{q}_j, \|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2), \quad \boldsymbol{v}_i = (\boldsymbol{v}, \boldsymbol{\sigma}, \boldsymbol{n}). \tag{4.4}$$

Database and Hyperparameters

The prismatic beam dimensions are $H = 10$, $W = 10$ and $L = 40$, discretized in $N_e = 500$ hexahedral linear brick elements and $N = 756$ nodes. The material hyperelastic and viscoelastic parameters are $C_{10} = 1.5 \cdot 10^5$, $C_{01} = 5 \cdot 10^3$, $D_1 = 10^{-7}$ and $\bar{g}_1 = 0.3$, $\bar{g}_2 = 0.49$, $\tau_1 = 0.2$, $\tau_2 = 0.5$ respectively. A distributed load of $F = 10^5$ is applied in $N_{\text{sim}} = 52$ different positions with an orientation perpendicular to the solid surface. The quasi-static simulation is discretized in $N_T = 20$ time increments of $\Delta t = 5 \cdot 10^{-2}$.

Following Eq. (4.4), the dimensions of the graph feature vectors are $F_e = 4$, $F_v = 11$ and $F_g = 0$. The hidden dimension of the node and edge latent vectors is $F_h = 50$. The learning rate is set to $l_r = 10^{-4}$ with decreasing order of magnitude on epochs

600 and 1200, and a total number of $N_{\text{epoch}} = 1800$. The training noise variance is set to $\sigma^2_{\text{noise}} = 10^{-5}$.

Results

The rollout results for the bending viscoelastic beam are presented in Fig. 4.9. The errors achieved by the present approach are again below the other two methods. The beam deformed configuration of three different test simulation snapshots are represented in Fig. 4.4, with the color code representing the xx component of the stress tensor. Similarly to the previous case, Fig. 4.7 (center) shows the thermodynamical consistency of our dynamical integration.

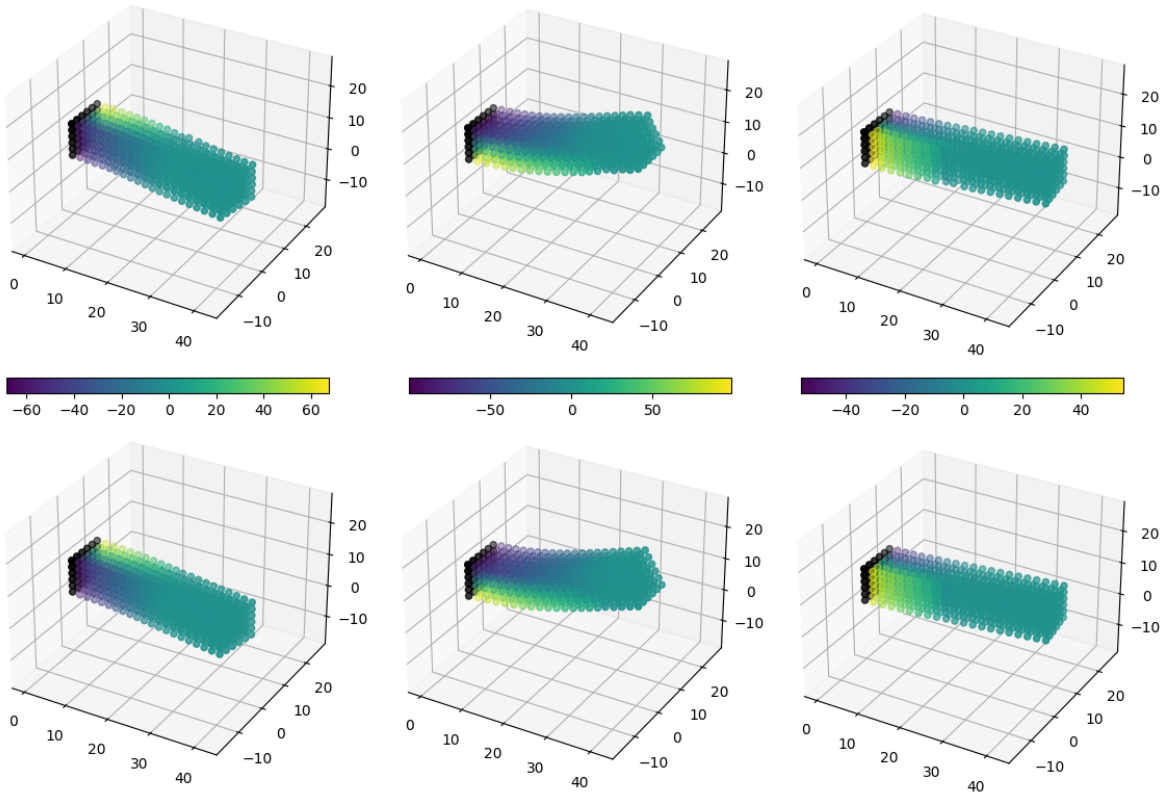


Figure 4.4: Top: Representation of a snapshot of three test simulations, i.e. not seen by the network on training, of the bending beam problem. Bottom: Their respective ground truth simulations. The color code represents the xx component of the dimensionless stress tensor, scaled $\times 0.001$.

4.3.3 Flow Past a Cylinder

Description

The last example consists of a viscous unsteady flow past a cylinder obstacle. The flow conditions are set to obtain varying Reynolds regimes, which result in Kármán vortex street and therefore a periodic behaviour in the steady state, see Fig. 4.5. The

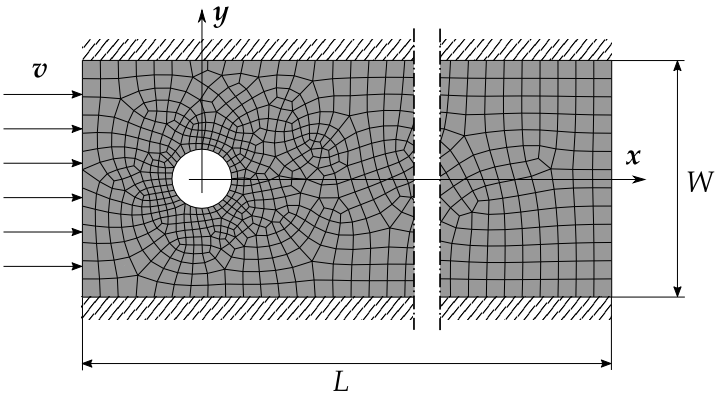


Figure 4.5: Unsteady flow past a cylinder obstacle. The flow velocity and cylinder obstacle position are varied to obtain different Reynolds numbers and flow profiles.

state variables for the flow past a cylinder are the velocity $\boldsymbol{v}$ and the pressure field P,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{v}, P) \in \mathbb{R}^2 \times \mathbb{R}\}.$$

The flow is computed with an Eulerian description of the output fields. Thus, the nodal coordinates ($\boldsymbol{q}^0$) are fixed in space and considered as edge features, whereas the whole state variables are assigned to the node features. An additional one-hot vector $\boldsymbol{n}$ is added to the node features in order to represent the inlet/outlet, walls or fluid nodes. No global feature vector is needed in this case, resulting in the following feature vectors:

$$\boldsymbol{e}_{ij} = (\boldsymbol{q}_i^0 - \boldsymbol{q}_j^0, \|\boldsymbol{q}_i^0 - \boldsymbol{q}_j^0\|_2), \quad \boldsymbol{v}_i = (\boldsymbol{v}, P, \boldsymbol{n}). \tag{4.5}$$

Database and Hyperparameters

The ground truth simulations are computed solving the 2D Navier Stokes equations. Six different obstacle positions are simulated with varying fluid discretization, which consist of approximately $N_e = 1100$ quadrilateral elements and $N = 1200$ nodes. No-slip conditions are forced in the stream walls and the cylinder obstacle. The fluid has a density of $\rho = 1$ and a dynamic viscosity of $\mu = 10^{-3}$. The variable freestream velocity is contained within the interval $\boldsymbol{v} \in [1, 2]$, summing a total of $N_{\text{sim}} = 30$ cases. The unsteady simulation is discretized in $N_T = 300$ time increments of $\Delta t = 10^{-2}$.

Following Eq. (4.5), the dimensions of the graph feature vectors are $F_e = 3$, $F_v = 8$ and $F_g = 0$. The hidden dimension of the node and edge latent vectors is $F_h = 128$. The learning rate is set to $l_r = 10^{-4}$ with decreasing order of magnitude on epochs 600 and 1200, and a total number of $N_{\text{epoch}} = 2000$. The training noise variance is set to $\sigma^2_{\text{noise}} = 4 \cdot 10^{-4}$.

Results

The rollout results for the flow past a cylinder problem are presented in Fig. 4.10. In this example the domain varies significantly, using a different unstructured mesh for each simulation. Thus, the graph-based architectures outperform the vanilla SPNN, which is meant for fixed structured problems. Considering the other two methods, our new approach outperforms the standard GNN architecture due to the metriplectic structure imposition over the dynamical problem, as depicted in Fig. 4.7. A single snapshot of the whole rollout of three different test simulations are represented in Fig. 4.6, with the color code representing the x component of the velocity field.

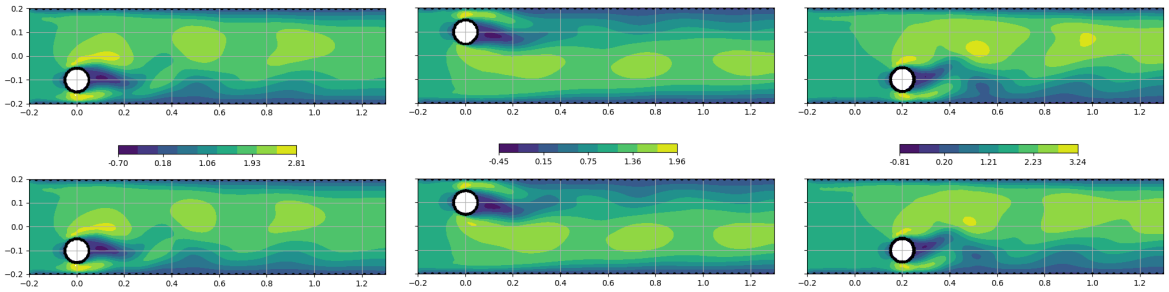


Figure 4.6: Top: Representation of a snapshot of three test simulations, i.e. not seen by the network on training, of the cylinder flow problem. Bottom: Their respective ground truth simulations. The color code represents the x component of the dimensionless velocity field.

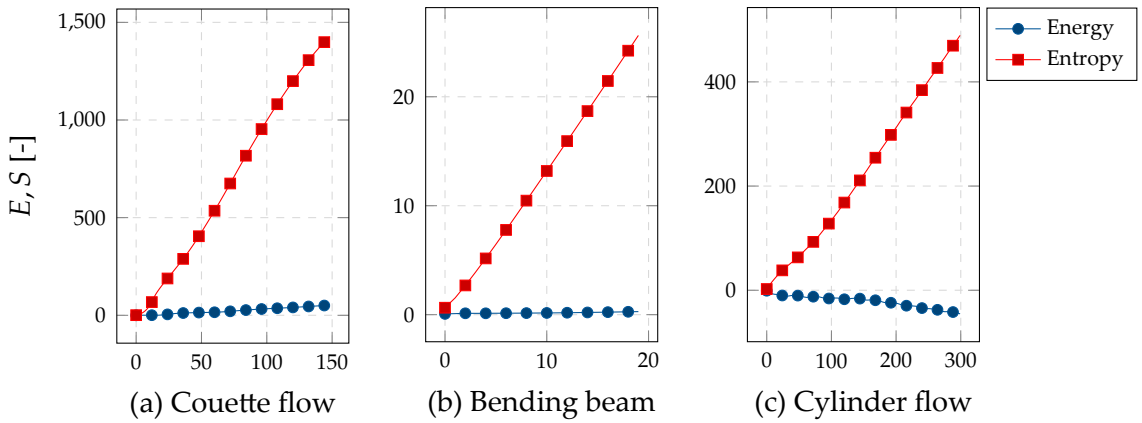


Figure 4.7: Conservation of energy and non-decreasing entropy potentials for a test case of the (a) Couette flow, (b) bending beam and (c) cylinder flow. Both quantities are averaged across all graph nodes for visualization.

4.4 Conclusions

We have presented a method to predict the time evolution of an arbitrary dynamical system based on two inductive biases. The metriplectic bias ensures the correct thermodynamic structure of the integrator based on the GENERIC formalism, whose

operators and potentials are estimated using computations over graphs, i.e. exploiting the geometric structure of the problem. The results show relative mean errors of less than 3% in all the tested examples, outperforming two other state-of-the-art techniques based on only physics-informed and geometric deep learning respectively.

These results confirm that both biases are necessary to achieve higher precision in the predicted simulations. The use of both techniques combines the computational power of geometric deep learning with the rigorous foundation of the GENERIC formalism, which ensures the thermodynamical consistency of the results.

The limitations of the presented technique are related to the computational complexity of the model. Large simulations with fine grids require a high amount of message passing to get the information across the whole domain, or a very fine time discretization, which both result in a high computational cost. Similarly, high-speed phenomena in relation to the wave velocity of the medium might be impossible to model. In Chapter 8 we discuss different techniques which may overcome the presented limitations as future work.

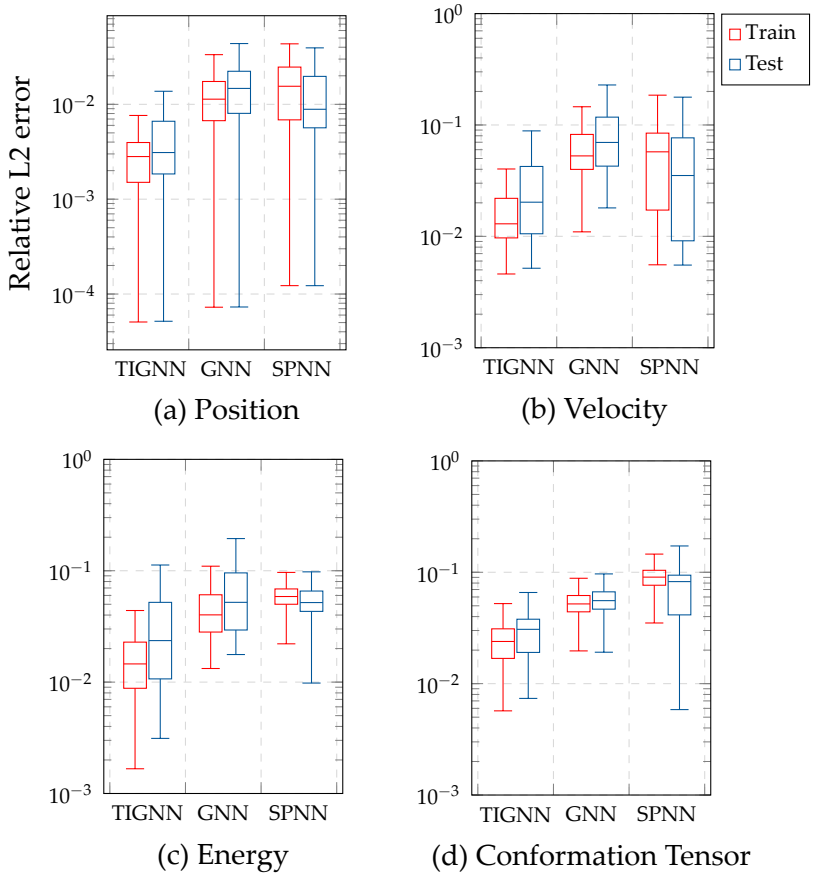


Figure 4.8: Box plots for the relative L2 error for all the rollout snapshots of the Couette flow in both train and test cases. The state variables represented are (a) position, (b) velocity, (c) energy and (d) conformation tensor.

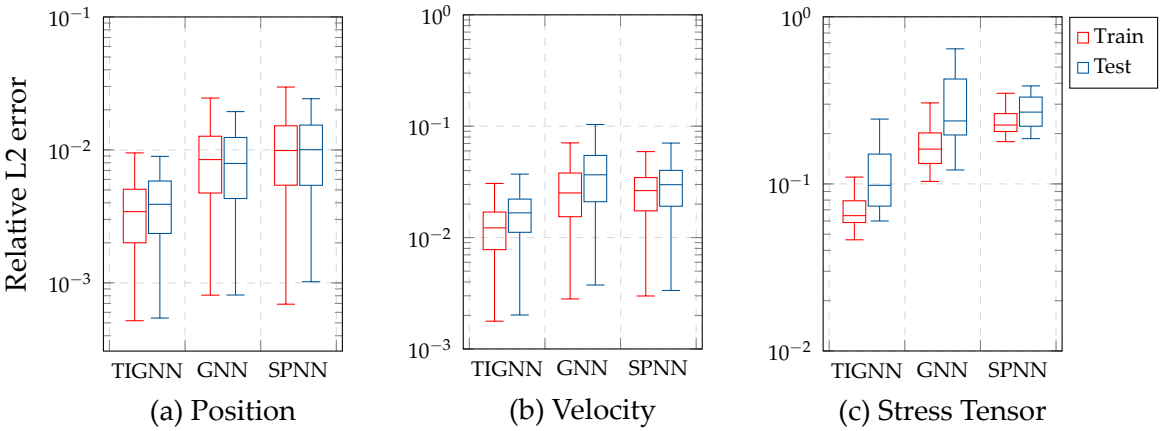


Figure 4.9: Box plots for the relative L2 error for all the rollout snapshots of the bending viscoelastic beam in both train and test cases. The state variables represented are (a) position, (b) velocity, and (c) stress tensor.

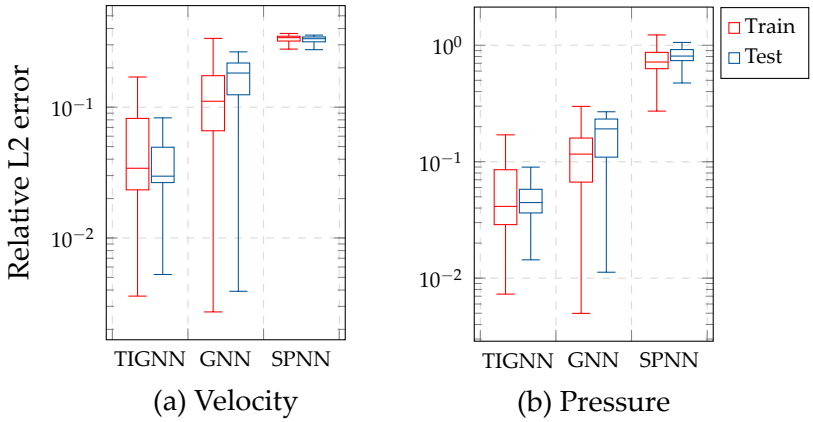


Figure 4.10: Box plots for the relative L2 error for all the rollout snapshots of the cylinder flow in both train and test cases. The state variables represented are (a) velocity and (b) pressure.

Part III

Latent Manifold Learning

Thermodynamics-Aware Model Order Reduction

5

In this chapter, we present an algorithm to learn the relevant latent variables of a large-scale discretized physical system and predict its time evolution using thermodynamically-consistent deep neural networks. Our method relies on sparse autoencoders, which reduce the dimensionality of the full order model to a set of sparse latent variables with no prior knowledge of the coded space dimensionality. Then, a second neural network is trained to learn the metriplectic structure of these reduced physical variables and predict its time evolution with a structure-preserving neural network presented in Chapter 3. The integrated paths can then be decoded to the original full-dimensional manifold and be compared to the ground truth solution. This method is tested with two examples applied to fluid and solid mechanics. The content of this chapter is included in the following publication [95]:

Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
Deep learning of thermodynamics-aware reduced-order models from data
Computer Methods in Applied Mechanics and Engineering, 379, 113763 (2021)

5.1 Introduction

Physical simulation has become an indispensable tool for engineers to recreate the operative conditions of a mechanical system and make decisions about its optimal design, ranging from composite building structures to complex fluid-solid interaction CFD simulations. These phenomena are often discretized in fine meshes resulting in millions of degrees of freedom, which are computationally expensive to handle, but their solutions are contained in lower-dimensional spaces. This is the so-called *manifold hypothesis* [57].

Thus, several methods try to overcome this inconvenience by reducing the dimensionality of the problem, computing a suitable reduced basis and projecting the full order model on it. The very first projection-based model order reduction (MOR)

methods relied on linear transformations with some additional constraints, such as Proper Orthogonal Decomposition (POD) [194, 47], Reduced-Basis technique [210] or Galerkin projection [225, 55]. However, these linear mappings are only locally accurate, so they fail in modeling more complex nonlinear phenomena and sometimes require prior information about the governing equations of the problem physics.

In order to overcome these limitations, several techniques have been developed in the machine learning framework that provide nonlinear mappings, such as Locally Linear Embedding (LLE) [10], Topological Data Analysis (TDA) [186], kernel Principal Component Analysis (k-PCA) [187] or Neural Networks, by means of Autoencoders [76]. In the present work we focus on this last method, which has proven to learn highly nonlinear manifolds in a wide variety of fields such as physics [54], chemistry [153], mechanics [144] or computational imaging [170]. Autoencoders used as a model reduction tool, project the original data (assumed to form a high-order manifold) to a reduced manifold. However, most of the current works rely on prior knowledge, or parametric search, of the optimal latent dimensionality of the problem. Here lies one of the key concepts of our method, which is able to learn a sparse representation of the latent space within a given reconstruction error bound.

The networks presented in Chapter 3 result in a thermodynamically-consistent integrator that is valid for both conservative (Hamiltonian) and dissipative systems. However, these networks operate only on full-order descriptions of the system, resulting in a costly procedure with limited engineering applicability for systems of tens of thousands to millions of degrees of freedom. The aim of this work is to apply this algorithm to more complex dynamical systems, combined with the nonlinear model order reduction power of autoencoders. The proposed methodology is a completely general method that is able to unveil the true effective dimensionality of the sampled data with no user intervention, and to construct from it a reduced-order integrator of the dynamics of the system with no previous knowledge on the nature of the system at hand. The resulting full-order reconstructions of the dynamics are guaranteed to conserve energy and dissipate entropy, as dictated by the laws of thermodynamics.

5.2 Problem Statement

We again consider a system whose governing variables will be hereafter denoted by $\boldsymbol{z} \in \mathcal{M} \subseteq \mathbb{R}^D$, with $\mathcal{M}$ the state space of these variables, which is assumed to have the structure of a differentiable manifold in $\mathbb{R}^D$. The full-order model of a given physical phenomenon can be expressed as a system of differential equations encoding the time evolution of a set of governing variables $\boldsymbol{z}$ as in Eq. (3.1).

The dimensionality reduction technique, in addition, seeks a simplified representation of the full-order state vector $\boldsymbol{z}$ through a set of latent, reduced variables $\boldsymbol{x} \in \mathcal{N} \subseteq \mathbb{R}^d$ contained in a trial manifold with reduced dimensionality, lower than the original

space $\mathcal{M}$. The mapping between both spaces can be denoted by $\varphi : \mathcal{M} \subseteq \mathbb{R}^D \rightarrow \mathbb{R}^d$ with $d \ll D$. Similarly, the inverse mapping φ^{-1} allows to undo the transformation, returning to the original full-order space.

The goal of this chapter is to find the convenient mapping φ for a dynamical system governed by Eq. (3.1) in order to efficiently learn the underlying physics in the reduced space $\mathcal{N}$ and then predict its time evolution. The solution is forced to fulfil the basic thermodynamic requirements of energy conservation and entropy inequality restrictions via the GENERIC formalism.

5.3 Methodology

The proposed algorithm divides the problem in two main steps, sketched in Fig. 5.1. First, the full order model is encoded to a reduced manifold with a nonlinear mapping via an autoencoder [144]. This autoencoder learns a latent representation of a state vector of a physical system, in order to handle a wide amount of simulation data in a compact form. The full order simulation data presented in this work is generated in silico, but the same procedure could be applied to measured data in a real physical system.

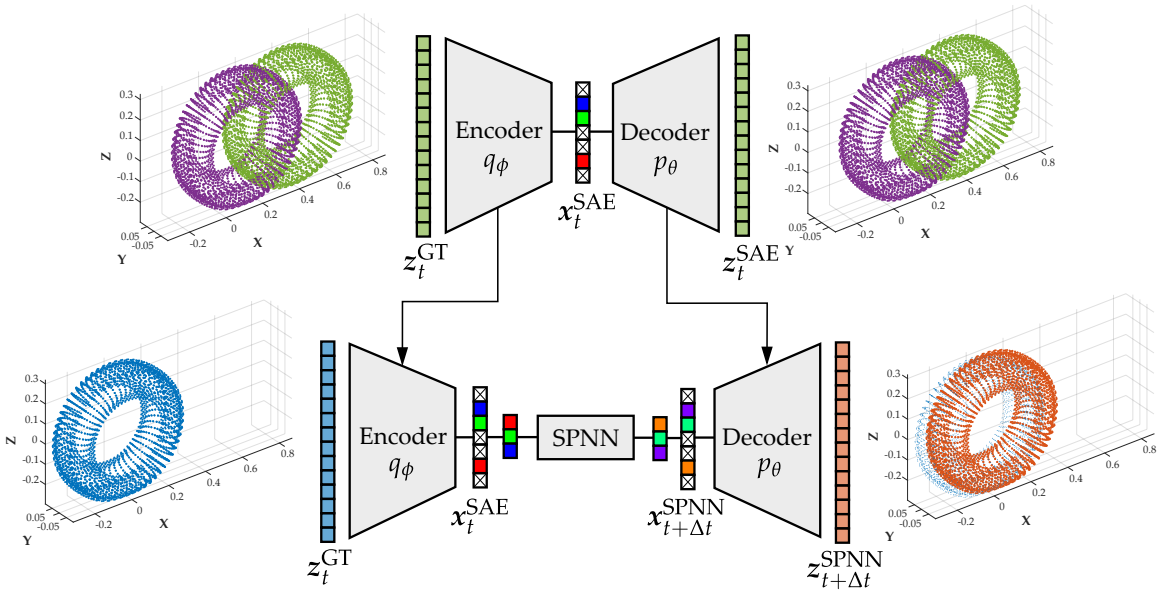


Figure 5.1: Block diagram of the proposed algorithm x_t^{SAE}. Snapshots of the rolling tire problem have been included for illustration purposes. Top: A sparse autoencoder (SAE) is trained with time snapshots of a ground truth physical simulation, in order to learn an encoded representation of the full-order space. Bottom: A structure-preserving neural network (SPNN) is trained to integrate the full time evolution of the latent variables, consistently with the GENERIC structure of the underlying physics of the problem.

Secondly, a structure-preserving neural network from Chapter 3 is trained with several snapshots of the physical simulation. This integration scheme preserves the thermodynamic structure of the latent variables in the reduced manifold [199] ensuring the basic laws of thermodynamics of energy conservation and entropy inequality. These integrated variables are then projected back to the original manifold of the full order model with the decoder, as depicted in Fig. 5.1.

5.3.1 Model Reduction with Sparse-Autoencoders

As already explained in Section 2.1, an autoencoder is composed by an encoder q_ϕ, which maps high-dimensional data $\boldsymbol{z} \in \mathbb{R}^D$ onto a low-dimensional code $\boldsymbol{x} \in \mathbb{R}^d$ with $d \ll D$, and a decoder p_θ, which applies the inverse mapping back to the original full-order manifold,

$$\begin{aligned} q_\phi &: \mathbb{R}^D \to \mathbb{R}^d, \quad \boldsymbol{x} = q_\phi(\boldsymbol{z}), \\ p_\theta &: \mathbb{R}^d \to \mathbb{R}^D, \quad \hat{\boldsymbol{z}} = p_\theta(\boldsymbol{x}). \end{aligned}$$

The vector $\boldsymbol{z}$ will be referred to as the *full order vector*, whereas its coded vector $\boldsymbol{x}$ is the *latent variable*. In this work, we use a bottleneck architecture composed by several stacked fully-connected hidden layers N_h in both the encoder and decoder. Each layer is modelled as a multilayer perceptron (MLP), defined as Eq. (2.1).

The latent vector dimensionality d in Eq. (5.1) is, a priori, unknown. Thus, we add a sparsity condition to the bottleneck to force the autoencoder to learn the number of latent variables needed to encode the necessary information of the full order model. Even if the latent layer has a fixed number of units N_d, the sparsity penalizer is able to find (at least a good approximation to) the intrinsic dimensionality of the low-dimensional data $\boldsymbol{x}$. Here, no prior on the reduced dimension is needed. Thus, further in this text, the autoencoder with sparsity regularization is referred as sparse autoencoder (SAE).

The loss function for our neural network is composed of two different terms:

- **Reconstruction loss**: This term minimizes the difference between the ground truth vector $\boldsymbol{z}_t^{\text{GT}}$ and the autoencoder reconstruction $\boldsymbol{z}_t^{\text{SAE}}$ in the snapshot n. This enforces the network to learn the identity function,

$$\mathcal{L}^{\text{rec}} = \left\| \boldsymbol{z}_t^{\text{GT}} - \boldsymbol{z}_t^{\text{SAE}} \right\|_2^2. \tag{5.3}$$

- **Regularization**: In order to impose the sparsity of the latent vector, several regularizers can be used [192]. Due to the continuous nature of the physical data, it is found more convenient to use L1-norm penalizer, which enforces

hard zeros in the latent variables that are not relevant,

$$\mathcal{L}^{\text{reg}} = \sum_{i=1}^{N_d} |\boldsymbol{x}_i^{\text{SAE}}|. \tag{5.4}$$

The temporal snapshots of the physical simulations are split in a partition of train snapshots ($N_{\text{train}} = 80\%$ of the database snapshots) and test snapshots ($N_{\text{test}} = 20\%$ of the database snapshots) so that $N_T = N_{\text{train}} + N_{\text{test}}$. The total loss function is computed as the mean squared error (MSE) of the data reconstruction loss and the sparsity regularization term for the train snapshots. The sparsity loss is multiplied by a regularization hyperparameter λ_r^{SAE}, which is responsible for the trade-off between the reconstruction fidelity of the autoencoder and the sparsity of the latent vector $\boldsymbol{x}$,

$$\mathcal{L}^{\text{SAE}} = \frac{1}{N_{\text{train}}} \sum_{n=0}^{N_{\text{train}}} (\mathcal{L}_n^{\text{rec}} + \lambda_r^{\text{SAE}} \mathcal{L}_n^{\text{reg}}). \tag{5.5}$$

The backpropagation algorithm [204] is then used to calculate the gradient of the loss function for each encoder and decoder parameters ϕ and θ (weight and bias vectors of both blocks), which are updated with the gradient descent technique [226]. An overview of the training and testing algorithms of the SAE are sketched in Algorithm 3 and Algorithm 4.

Algorithm 3: Pseudocode for the training algorithm of the Sparse-Autoencoder.

Load database: z^{GT} (train partition);
Define network architecture: $N_{\text{in}}^{\text{SAE}} = N_{\text{out}}^{\text{SAE}} = D$, N_h^{SAE}, N_d^{SAE}, σ_j^{SAE};
Define hyperparameters: l_r^{SAE}, λ_r^{SAE};
Initialize $w_{i,j}^{\text{SAE}}$, b_j^{SAE};
for *epoch* $\leftarrow 1 : N_{\text{epoch}}$ **do**
  Initialize loss function: $C = 0$;
  for *train_case* $\leftarrow 1 : N_{\text{train}}$ **do**
    Encoder: $\boldsymbol{x}_t^{\text{SAE}} = q_\phi(\boldsymbol{z}_t^{\text{GT}})$; ▷ Eq. (5.1)
    Decoder: $\boldsymbol{z}_t^{\text{SAE}} = p_\theta(\boldsymbol{x}_t^{\text{SAE}})$; ▷ Eq. (5.2)
    Loss function: $C \leftarrow C + \mathcal{L}^{\text{rec}} + \lambda_r^{\text{SAE}} \mathcal{L}^{\text{reg}}$; ▷ Eq. (5.3)–(5.4)
  end for
  MSE loss function: $\mathcal{L}^{\text{SAE}} \leftarrow \frac{C}{N_{\text{train}}}$ ▷ Eq. (5.5)
  Backward propagation;
  Optimizer step;
end for

Once the problem is reduced to a lower-dimensional manifold, a second neural network can be trained to learn the underlying physics of the problem, being able to integrate the whole simulation trajectory with thermodynamic consistency.

Algorithm 4: Pseudocode for the test algorithm of the Sparse-Autoencoder.

Load database: z^{GT} (test partition);
Load network parameters;
for *test_case* $\leftarrow 1 : N_{\text{test}}$ **do**
  Encoder: $x_t^{\text{SAE}} = q_\phi(z_t^{\text{GT}})$; ▷ Eq. (5.1)
  Decoder: $z_t^{\text{SAE}} = p_\theta(x_t^{\text{SAE}})$; ▷ Eq. (5.2)
  Compute Squared Error: $\varepsilon^2 = \left(z_t^{\text{GT}} - z_t^{\text{SAE}}\right)^2$; ▷ Eq. (5.11)
end for
Compute MSE^{SAE} (z); ▷ Eq. (5.11)

5.3.2 Structure-Preserving Neural Networks

We use a structure-preserving neural network (SPNN) presented in Chapter 3 to impose the GENERIC structured in the discretized approach,

$$x_{t+\Delta t} = x_t + \Delta t \left(L \frac{\mathrm{D}E}{\mathrm{D}x_t} + M \frac{\mathrm{D}S}{\mathrm{D}x_t} \right), \tag{5.6}$$

subject to

$$L \frac{\mathrm{D}S}{\mathrm{D}x_t} = 0, \quad M \frac{\mathrm{D}E}{\mathrm{D}x_t} = 0,$$

ensuring the thermodynamical consistency of the resulting model. From now on, the energy and entropy gradients will be shortened as $\frac{\mathrm{D}E}{\mathrm{D}x_t} \equiv \mathrm{DE}$ and $\frac{\mathrm{D}S}{\mathrm{D}x_t} \equiv \mathrm{DS}$.

Unlike in Chapter 3, the GENERIC structure is imposed to the reduced order model [79] learnt by the sparse autoencoder, so there is no prior information about the L and M matrices. Instead, the SPNN is forced to automatically learn them on each learning set time step with their respective skew-symmetric and symmetric conditions. Similarly, the energy and entropy gradients are computed on each time step and no finite-difference approach is needed.

The input of the neural network is the encoded vector state of a given time step $x_t^{\text{SAE}} = q_\phi(z_t^{\text{GT}})$, and the outputs are the concatenated GENERIC matrices (L, M) and energy and entropy gradient matrices (DE, DS). Then, using the GENERIC forward integration scheme in Eq. (5.6), the reduced state vector at the next time step $x_{t+\Delta t}^{\text{SPNN}}$ is obtained.

Thus, the input dimension of the SPNN is the same as the dimension of the sparsified latent variables x_t^{SAE} ($N_{\text{in}}^{\text{SPNN}} = d$). Consequently, the GENERIC matrices L and M are squared with dimension d^2 each, which can be reduced to $d(d-1)/2$ and $d(d+1)/2$ taking into account the skew-symmetric and symmetric elements respectively. Additionally, the matrix M is assembled by taking the absolute value of the diagonal elements of the resulting lower triangular matrix and multiplying it by its transpose. By the Cholesky factorization, this ensures that M is positive semidefinite. The energy and entropy gradient matrices DE and DS have the same dimension

d as the state vector. The final output dimension of the integrator network is then $N_{\text{out}}^{\text{SPNN}} = d(d+1)/2 + d(d-1)/2 + 2d = d(d+2)$.

The loss function for the SPNN is composed of three different terms:

- **Data loss**: The main loss condition is the agreement between the network output and the real data. It is computed as the squared error sum, computed between the predicted state vector $\boldsymbol{x}_{t+\Delta t}^{\text{SPNN}}$ and the ground truth solution based on the SAE output $\boldsymbol{x}_{t+\Delta t}^{\text{SAE}}$ for each time step,

$$\mathcal{L}^{\text{data}} = \left\| \boldsymbol{x}_{t+\Delta t}^{\text{SAE}} - \boldsymbol{x}_{t+\Delta t}^{\text{SPNN}} \right\|_2^2. \tag{5.7}$$

- **Fulfillment of the degeneracy conditions**: The loss function will also account for the degeneracy conditions in order to ensure the thermodynamic consistency of the solution, implemented as the sum of the squared elements of the degeneracy vectors for each time step,

$$\mathcal{L}^{\text{deg}} = \|\boldsymbol{L}\text{DS}\|_2^2 + \|\boldsymbol{M}\text{DE}\|_2^2. \tag{5.8}$$

  This term acts as a regularization of the loss function and, at the same time, is the responsible of ensuring thermodynamic consistency of the integration scheme.

- **Regularization**: In order to avoid overfitting, an extra L2 regularization term $\mathcal{L}^{\text{reg}}$ is added to the loss function,

$$\mathcal{L}^{\text{reg}} = \sum_{l}^{L} \sum_{i}^{n^{[l]}} \sum_{j}^{n^{[l+1]}} (w_{i,j}^{[l],\text{SPNN}})^2. \tag{5.9}$$

The same database split procedure is followed as in the SAE, dividing the complete dataset of N_T snapshots in a partition of train snapshots ($N_{\text{train}} = 80\%$ of the database snapshots) and test snapshots ($N_{\text{test}} = 20\%$ of the database snapshots) so that $N_T = N_{\text{train}} + N_{\text{test}}$. The total loss function is computed as the mean squared error (MSE) of the data loss and degeneracy residual, in addition to the regularization term, for all the training snapshots of the simulation time T. Both the data loss error and the regularization terms are weighted with two additional hyperparameters λ_d^{SPNN} and λ_r^{SPNN} respectively, which account for their relative influence in the total loss function with respect to the degeneracy constraint,

$$\mathcal{L}^{\text{SPNN}} = \frac{1}{N_{\text{train}}} \sum_{n=0}^{N_{\text{train}}} (\lambda_d^{\text{SPNN}} \mathcal{L}_n^{\text{data}} + \mathcal{L}_n^{\text{deg}}) + \lambda_r^{\text{SPNN}} \mathcal{L}^{\text{reg}}. \tag{5.10}$$

The usual backpropagation algorithm [204] is then used to calculate the gradient of the loss function for each net parameter (weight and bias vectors), which are updated with the gradient descent technique [226]. The training algorithm is sketched below in Algorithm 5.

Algorithm 5: Pseudocode for the train algorithm of the latent SPNN.

Load train database: z^{SAE} (train partition), Δt;
Define network architecture: $N_{\text{in}}^{\text{SPNN}} = d$, $N_{\text{out}}^{\text{SPNN}} = d(d+3)$, N_h^{SPNN}, σ_j^{SPNN};
Define hyperparameters: l_r^{SPNN}, λ_d^{SPNN}, λ_r^{SPNN};
Initialize $w_{i,j}^{\text{SPNN}}$, b_j^{SPNN};
for $epoch \leftarrow 1 : N_{\text{epoch}}$ **do**
  Initialize loss function: $C = 0$;
  for $train\_case \leftarrow 1 : N_{\text{train}}$ **do**
    Encoder: $\boldsymbol{x}_t^{\text{SAE}} = q_\phi(\boldsymbol{z}_t^{\text{GT}})$; ▷ Eq. (5.2)
    Forward propagation: $[\boldsymbol{L}, \boldsymbol{M}, \text{DE}, \text{DS}] \leftarrow \text{SPNN}(\boldsymbol{x}_t^{\text{SAE}})$; ▷ Eq. (2.1)
    Time step integration: $\boldsymbol{x}_{t+\Delta t}^{\text{SPNN}} \leftarrow \boldsymbol{x}_t^{\text{SAE}} + \Delta t\ (\boldsymbol{L}\text{DE} + \boldsymbol{M}\text{DE})$; ▷ Eq. (5.6)
    Update loss function: $C \leftarrow C + \lambda_d^{\text{SPNN}} \mathcal{L}^{\text{data}} + \mathcal{L}^{\text{deg}}$; ▷ Eq. (5.7)–(5.8)
  end for
  MSE loss function: $\mathcal{L}^{\text{SPNN}} \leftarrow \frac{C}{N_{\text{train}}} + \lambda_r^{\text{SPNN}} \mathcal{L}^{\text{reg}}$ ▷ Eq. (5.9)–(5.10)
  Backward propagation;
  Optimizer step;
end for

The testing consists of the full time integration of the initial state vector $\boldsymbol{z}_0$ at $t = 0$ along the complete simulation time interval $\mathcal{I} = (0, T]$, reproducing the problem statement established in Eq. (3.1). A pseudocode of the testing process of the SPNN is shown in Algorithm 6.

Algorithm 6: Pseudocode for the test algorithm of the latent SPNN.

Load database: z^{GT}, Δt;
Load network parameters;
Initialize state vector: $\boldsymbol{z}_0^{\text{SAE}} = \boldsymbol{z}_0^{\text{SPNN}} = \boldsymbol{z}_0^{\text{GT}}$;
Initialize encoded state vector: $\boldsymbol{x}_0^{\text{SAE}} = \boldsymbol{x}_0^{\text{SPNN}} = q_\phi(\boldsymbol{z}_0^{\text{GT}})$; ▷ Eq. (5.1)
for $snapshot \leftarrow 1 : N_T$ **do**
  Forward propagation: $[\boldsymbol{L}, \boldsymbol{M}, \text{DE}, \text{DS}] \leftarrow \text{SPNN}(\boldsymbol{x}_t^{\text{SPNN}})$; ▷ Eq. (2.1)
  Time step integration: $\boldsymbol{x}_{t+\Delta t}^{\text{SPNN}} \leftarrow \boldsymbol{x}_t^{\text{SPNN}} + \Delta t\ (\boldsymbol{L}\text{DE} + \boldsymbol{M}\text{DE})$; ▷ Eq. (5.6)
  Update state vector: $\boldsymbol{x}_t^{\text{SPNN}} \leftarrow \boldsymbol{x}_{t+\Delta t}^{\text{SPNN}}$;
  Decoder: $\boldsymbol{z}_{t+\Delta t}^{\text{SPNN}} = p_\theta(\boldsymbol{x}_{t+\Delta t}^{\text{SPNN}})$; ▷ Eq. (5.2)
  Compute Squared Error: $\varepsilon^2 = \left(\boldsymbol{z}_{t+\Delta t}^{\text{GT}} - \boldsymbol{z}_{t+\Delta t}^{\text{SPNN}}\right)^2$; ▷ Eq. (5.12)
end for
Compute $\text{MSE}^{\text{SPNN}}(\boldsymbol{z})$; ▷ Eq. (5.12)

5.3.3 Evaluation Metrics

The SAE performance is then evaluated with the mean squared error (MSE) of the test snapshots for each state variable (z),

$$\text{MSE}^{\text{SAE}}\ (z) = \frac{1}{N_{\text{test}}} \sum_{n=0}^{N_{\text{test}}} \varepsilon_n^2 = \frac{1}{N_{\text{test}}} \sum_{n=0}^{N_{\text{test}}} \left(z_t^{\text{GT}} - z_t^{\text{SAE}} \right)^2, \tag{5.11}$$

tested with two different databases of nonlinear systems. Then, the integration performance is evaluated with the mean squared error (MSE) of the SPNN state variable predictions and the ground truth solution for the complete set of snapshots N_T,

$$\text{MSE}^{\text{SPNN}}\ (z) = \frac{1}{N_T} \sum_{n=0}^{N_T} \varepsilon_n^2 = \frac{1}{N_T} \sum_{n=0}^{N_T} \left(z_t^{\text{GT}} - z_t^{\text{SPNN}} \right)^2, \tag{5.12}$$

tested for the same nonlinear systems trained in the SAE training phase.

The SPNN is compared on each example with a baseline unconstrained neural network which directly predicts the time evolution of the latent vector $\boldsymbol{x}_{t+\Delta t}$ from the current snapshot $\boldsymbol{x}_t$, with a similar training and integration scheme as depicted in Algorithm 5 and Algorithm 6.

5.4 Experiments

5.4.1 Couette Flow of an Oldroyd-B Fluid

Description

The first example is a shear (Couette) flow of an Oldroyd-B fluid model presented in Chapter 3, see Fig. 3.7. The state variables chosen for the full order model are the position of the fluid on each node of the mesh $\boldsymbol{q}$, its velocity $\boldsymbol{v}$ in the x direction, internal energy e and the conformation tensor shear component τ for all the nodes of the mesh,

$$\mathcal{S} = \{ \boldsymbol{z} = (\boldsymbol{q}_i, \boldsymbol{v}_i, e_i, \tau_i, i = 1, 2, ..., N) \in (\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R})^N \}, \tag{5.13}$$

resulting in a full-order model of $D = 4N$ dimensions.

Database and Hyperparameters

The training database for this Oldroyd-B model is generated in MATLAB with a multiscale approach [139] in dimensionless form. The fluid is discretized in the vertical direction with $N = 100$ elements (101 nodes) in a total height of $H = 1$. A total of

10^4 dumbells were considered at each nodal location in the model. The lid velocity is set to $V = 1$, the viscolastic Weissenberg number We $= 1$ and Reynolds number of Re $= 0.1$. The simulation time of the movement is $T = 1$ in time increments of $\Delta t = 0.0067$ ($N_T = 150$ snapshots).

The database consists of the state vector, Eq. (5.13), of the 100 nodal trajectories (excluding the node at $y = H$, for which a no-slip condition $v = 0$ has been imposed) for each snapshot of the simulation. This database is split in 120 train snapshots and 30 test snapshots.

The SAE input and output sizes are $N_{\text{in}}^{\text{SAE}} = N_{\text{out}}^{\text{SAE}} = D = 4N = 400$. The number of hidden layers in both the encoder and decoder is $N_h^{\text{SAE}} = 2$ with 160 neurons each, ReLU activation functions and linear in the first and last layer. The number of bottleneck variables is set to $N_d = 10$. It is initialized according to the Kaiming method [92], with normal distribution and the optimizer used is Adam [128], with a learning rate of $l_r^{\text{SAE}} = 10^{-4}$. The sparsity parameter is set to $\lambda_r^{\text{SAE}} = 10^{-4}$. The training process (Algorithm 3) is able to sparsify the bottleneck variables of the Oldroyd-B model with only $d = 4$ latent variables, which are the input variables used in the structure preserving-neural network.

Thus, the SPNN input and output size are $N_{\text{in}}^{\text{SPNN}} = d = 4$ and $N_{\text{out}}^{\text{SPNN}} = d(d + 2) = 24$. The number of hidden layers is $N_h^{\text{SPNN}} = 5$ with 24 neurons each, ReLU activation functions and linear in the last layer. The same initialization method and optimizer are used as in the SAE network, with a learning rate of $l_r^{\text{SPNN}} = 10^{-5}$. The weight decay and the data weight are set to $\lambda_r^{\text{SPNN}} = 10^{-5}$ and $\lambda_d^{\text{SPNN}} = 10^3$ respectively.

The unconstrained network training parameters are analogous to the SPNN, except for the output size $N_{out}^{UC} = N_{in}^{UC} = 4$. Several network architectures were tested, and the lowest error is achieved with $N_h = 5$ hidden layers and 25 neurons each layer.

Results

Fig. 5.2 shows the time evolution of the SAE bottleneck variables after the complete training process. The sparsity constraint forces the unnecessary latent variables to vanish, remaining a learnt latent dimensionality of $d = 4$ relevant variables from a starting bottleneck dimension of $N_d = 10$ (Fig. 5.2, top left). This compares advantageously with the obtained dimensionality $d = 6$ of our previous work [74]. Table 5.1 shows the mean squared error of the SAE reconstruction, computed with Algorithm 4, and an equal reduction using Proper Orthogonal Decomposition. Then, the SPNN is able to integrate the whole trajectory of the relevant latent variables in the reduced manifold in good agreement with the original SAE reduction (Fig. 5.2, top right). The integration scheme also ensures that the time derivative of the energy

($\dot{E}$) and entropy ($\dot{S}$) of the system remain equal to zero or greater than zero respectively, in fulfilment with the first and second laws of thermodynamics, see Fig. 5.2 ottom.

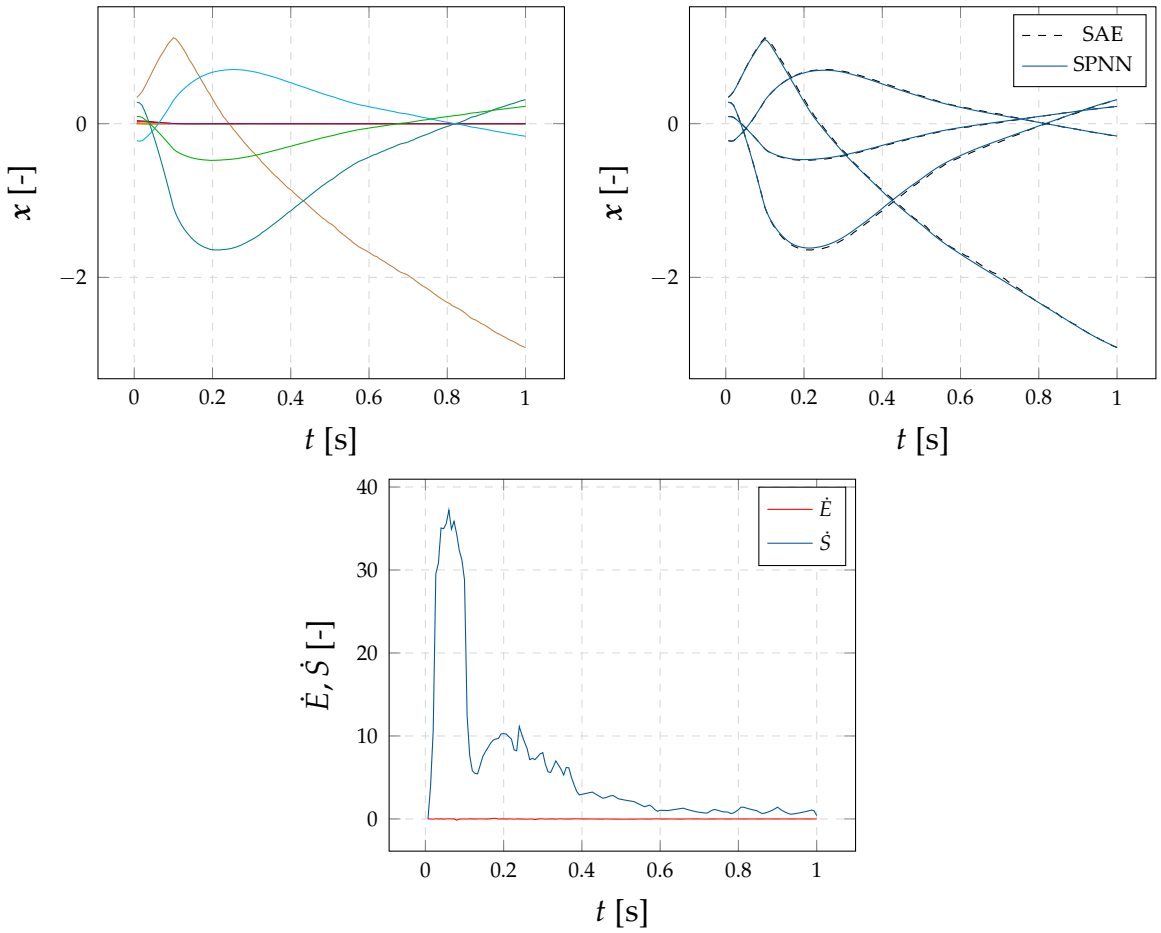


Figure 5.2: Top left: Time evolution of the latent variables encoded with the SAE in the Couette flow problem. The bottleneck has $N_d = 10$ neurons and the learning algorithm automatically sparsifies them to a dimensionality of $d = 4$ relevant latent variables. Top right: Time evolution of the relevant latent variables integrated in time by the SPNN. Bottom: Evolution of the time derivative of the energy ($\dot{E}$) and entropy ($\dot{S}$) of the latent variables.

Fig. 5.3 presents the time evolution of the decoded state variables of the Couette flow for 4 different nodes computed with the presented integration scheme and the ground truth. The results show a good agreement in the transient response of the Couette flow, even for the high nonlinearities of the internal energy and the conformation tensor shear component. The mean squared error of the total integration scheme, computed with Algorithm 6, for the 4 state variables is reported in Table 5.2 using the SPNN and the unconstrained approach (UC). Our neural network achieves less error than the unconstrained one, showing the importance of adding the physical constraints to the learning process, and this difference becomes greater in the second example.

Table 5.1: Mean squared error of the SAE reconstruction (MSE^{SAE}) and the same reduction using a POD algorithm (MSE^{POD}) for the Couette flow example, only for the test snapshots.

State variable (z_i)	MSE^{SAE}	MSE^{POD}
q [-]	$2.52 \cdot 10^{-6}$	$7.87 \cdot 10^{-6}$
v [-]	$7.27 \cdot 10^{-5}$	$4.31 \cdot 10^{-5}$
e [-]	$1.89 \cdot 10^{-6}$	$7.33 \cdot 10^{-6}$
τ [-]	$7.22 \cdot 10^{-6}$	$2.07 \cdot 10^{-5}$

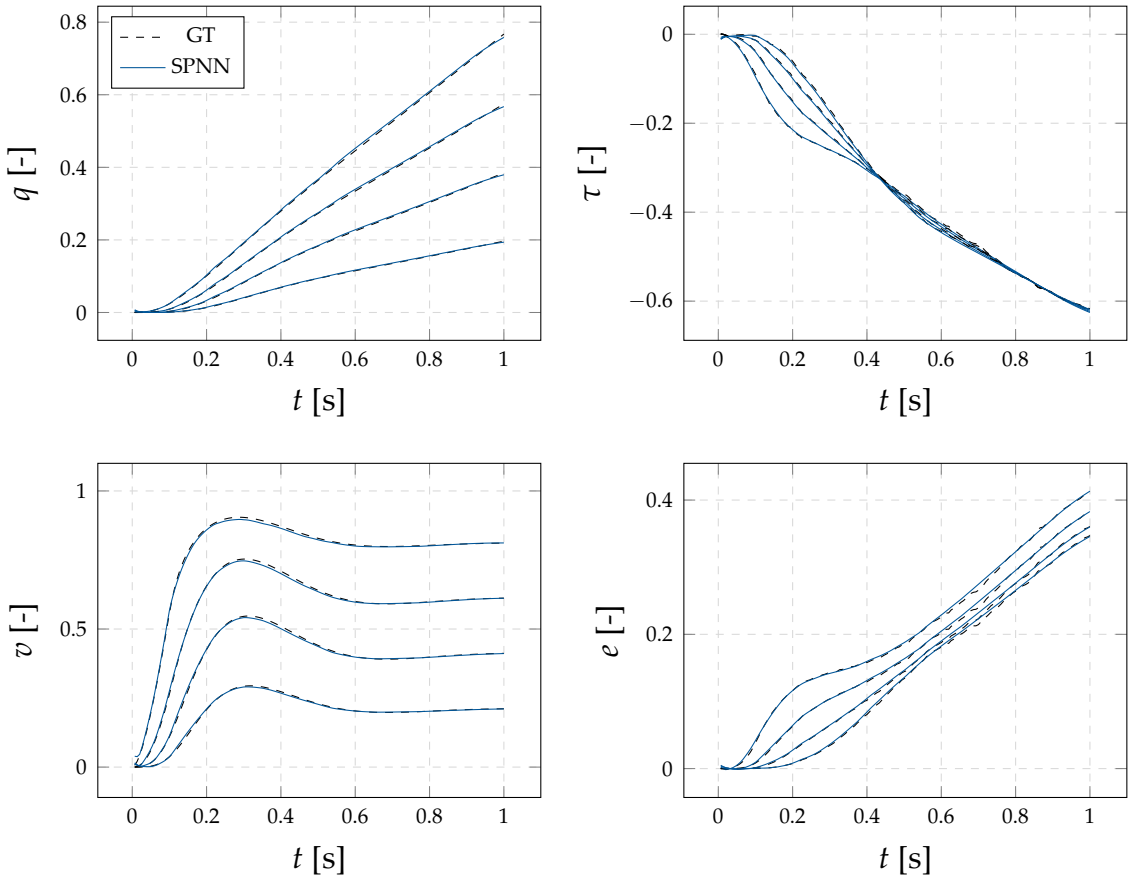


Figure 5.3: Results of the complete integration scheme (SPNN) with respect to the ground truth simulation (GT) for 4 different nodes of the Oldroyd-B fluid database.

Table 5.2: Mean squared error of the SPNN integration scheme and the unconstrained (UC) approach for the Couette flow example, reported for the complete trajectory.

State variable (z_i)	MSE^{SPNN}	MSE^{UC}
q [-]	$1.78 \cdot 10^{-5}$	$7.96 \cdot 10^{-5}$
v [-]	$3.34 \cdot 10^{-5}$	$3.48 \cdot 10^{-5}$
e [-]	$5.60 \cdot 10^{-6}$	$5.67 \cdot 10^{-5}$
τ [-]	$2.19 \cdot 10^{-5}$	$1.22 \cdot 10^{-4}$

5.4.2 Rolling Hyperelastic Tire

Description

The second example is a simulation of the transient response of a 175 SR14 rolling tire ($D_{\text{tire}} = 0.66$ m) impacting with a curb ($h_{\text{curb}} = 0.025$ m). The tire is initially preloaded with an inflation load of 200 kPa, simulating the internal air pressure, and a footprint load of 3300 N in the vertical axis, simulating the weight of the vehicle distributed equally in the tires. The free rolling conditions are determined in a separated analysis, corresponding to $\omega = 8.98$ rad/s for a translational horizontal velocity of $v_0 = 10$ km/h (see Fig. 5.4).

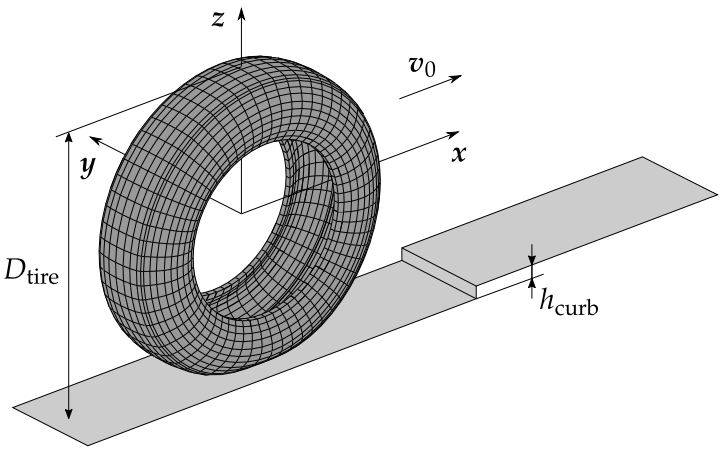


Figure 5.4: Hyperelastic tire rolling towards a curb. 3D position, 3D velocity and Cauchy stress tensor components are tracked for the total of 4140 selected nodes.

The tread and sidewalls of the tire are made of rubber, modeled as an incompressible hyperelastic material with a viscolastic component described by a one-term Prony series of the dimensionless shear relaxation modulus,

$$g_R(t) = 1 - \bar{g}_1(1 - e^{\frac{-t}{\tau_1}}),$$

with relaxation coefficient of $\bar{g}_1 = 0.3$ and relaxation time of $\tau_1 = 0.1$ s. The belts and carcass of the tire are constructed from fiber-reinforced rubber composites, modeled as a linear elastic material, with a 20° orientation of the reinforcing belt.

The state variables chosen for the full order model are the 3D position $\boldsymbol{q}_i$, velocity $\boldsymbol{v}_i$ and the 6 different components of the Cauchy stress tensor $\boldsymbol{\sigma}_i$ for each i node of the studied mesh subset N,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}_i, \boldsymbol{v}_i, \boldsymbol{\sigma}_i, i = 1, 2, ..., N) \in (\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^6)^N\},$$

resulting in a full-order model of $D = 12N$ dimensions.

Database and Hyperparameters

The training database for this rolling tire simulation is generated by finite element simulation. The full-order model is discretized with 5283 elements in a total of 6962 nodes. The simulation time of the movement is $T = 0.5$ s in time increments of $\Delta t = 0.0025$ s ($N_T = 200$ snapshots). The database consists of the normalized state vector of a subset of $N = 4140$ relevant nodes in every time step snapshot. The total state vector snapshots are randomly split in 160 train snapshots and 40 test snapshots.

The SAE architecture for this second example is slightly modified in order to handle the high dimensionality of the problem. The three physical variables ($\boldsymbol{q}$, $\boldsymbol{v}$, and $\boldsymbol{\sigma}$) are encoded and decoded independently, due to their very different nature. In this way, three bottleneck latent vectors are obtained. The input and output sizes of the three SAEs are $N^{\text{SAE}}_{\text{in},q} = N^{\text{SAE}}_{\text{out},q} = 3N = 12420$ for the position variable, $N^{\text{SAE}}_{\text{in},v} = N^{\text{SAE}}_{\text{out},v} = 3N = 12420$ and $N^{\text{SAE}}_{\text{in},\sigma} = N^{\text{SAE}}_{\text{out},\sigma} = 6N = 24840$ for the stress tensor.

The number of hidden layers in both the encoder and decoder is $N^{\text{SAE}}_h = 2$ in the three variables with 40 neurons each in position and velocity, and 80 neurons in the stress tensor, with ReLU activation functions and linear in the first and last layers. The number of bottleneck variables is set to $N_{d,q} = 10$ for the position, $N_{d,v} = 10$ for velocity and $N_{d,\sigma} = 20$ for the stress tensor. Thus, the total dimensionality of the bottleneck latent vector is $N_d = N_{d,q} + N_{d,v} + N_{d,\sigma} = 40$.

In the same way as we do in the first example, the nets are initialized according to the Kaiming method [92], with normal distribution and the optimizer used is Adam [128], with a learning rate of $l^{\text{SAE}}_r = 10^{-4}$. The sparsity parameter, in this case, is set to $\lambda^{\text{SAE}}_r = 10^{-2}$. The training process (Algorithm 3) is able to sparsify the bottleneck variables of the rolling tire model with only $d_q = 4$ position, $d_q = 3$ velocity and $d_\sigma = 2$ stress tensor latent variables. So, the learnt dimensionality of the reduced model is $d = d_q + d_v + d_\sigma = 9$, which are the input variables used in the structure preserving-neural network.

Thus, the SPNN input and output sizes are $N^{\text{SPNN}}_{\text{in}} = d = 9$ and $N^{\text{SPNN}}_{\text{out}} = d(d+2) = 99$. The number of hidden layers is $N^{\text{SPNN}}_h = 5$ with 198 neurons each, with ReLU activation functions and linear in the last layer. The same initialization method and optimizer are used as in the SAE network, with a learning rate of $l^{\text{SPNN}}_r = 10^{-5}$. The weight decay and the data weight are set to $\lambda^{\text{SPNN}}_r = 10^{-4}$ and $\lambda^{\text{SPNN}}_d = 10^3$ respectively.

The unconstrained network training parameters are analogous to the SPNN, except for the output size $N^{UC}_{out} = N^{UC}_{in} = 9$. Several network architectures were tested, and the lowest error i was achieved with $N_h = 5$ hidden layers and 45 neurons each layer.

Results

Fig. 5.5 shows the time evolution of the SAE bottleneck variables ($\boldsymbol{x}_q$, $\boldsymbol{x}_v$ and $\boldsymbol{x}_\sigma$) after the complete training process. The sparsity constraint forces the unnecessary latent variables to vanish, remaining a learnt latent dimensionality of $d_q = 4$, $d_v = 3$ and $d_\sigma = 2$ relevant variables from a starting bottleneck dimension of $N_{d,q} = 10$, $N_{d,v} = 10$ and $N_{d,\sigma} = 20$ respectively (Fig. 5.5). The mean squared error of the SAE reconstruction, computed with Algorithm 4, and a equal reduction with a Proper Orthogonal Decomposition is shown in Table 5.3. Then, the SPNN is able to integrate the whole trajectory of the relevant latent variables in the reduced manifold in good agreement with the original SAE reduction, see Fig. 5.5 middle right. Also, the integration scheme fulfils the first and second laws of thermodynamics, see Fig. 5.5 bottom.

Table 5.3: Left: Mean squared error of the SAE reconstruction (MSE^{SAE}) for the 12 state variables of the rolling tire example, reported only for the test snapshots. Right: Mean squared error of the same reduction using a Proper Orthogonal Decomposition algorithm (MSE^{POD}).

State variable (z_i)	MSE^{SAE}	MSE^{POD}
q_1 [m]	$2.37 \cdot 10^{-5}$	$1.30 \cdot 10^{-3}$
q_2 [m]	$3.69 \cdot 10^{-7}$	$6.27 \cdot 10^{-7}$
q_3 [m]	$3.06 \cdot 10^{-5}$	$6.55 \cdot 10^{-5}$
v_1 [m/s]	$1.00 \cdot 10^{-3}$	$3.32 \cdot 10^{-2}$
v_2 [m/s]	$4.54 \cdot 10^{-5}$	$2.37 \cdot 10^{-2}$
v_3 [m/s]	$3.70 \cdot 10^{-3}$	$6.91 \cdot 10^{-2}$
σ_{11} [MPa]	$2.41 \cdot 10^{-4}$	$3.74 \cdot 10^{-4}$
σ_{22} [MPa]	$2.10 \cdot 10^{-4}$	$4.34 \cdot 10^{-4}$
σ_{33} [MPa]	$3.35 \cdot 10^{-4}$	$6.40 \cdot 10^{-4}$
σ_{12} [MPa]	$6.73 \cdot 10^{-5}$	$1.17 \cdot 10^{-4}$
σ_{13} [MPa]	$1.80 \cdot 10^{-4}$	$3.24 \cdot 10^{-4}$
σ_{23} [MPa]	$2.95 \cdot 10^{-5}$	$5.86 \cdot 10^{-5}$

Fig. 5.6 presents the time evolution of the decoded state variables q_3, v_3, σ_{33} and σ_{23} of the rolling hyperelastic tire for 4 different nodes computed with the presented integration scheme and the ground truth. The mean squared error of the total integration scheme, computed with Algorithm 6, for the 12 state variables is reported in Table 5.4 using the SPNN and the unconstrained approach (UC). In this example, the error achieved by our method is several orders of magnitude less than the naive approach.

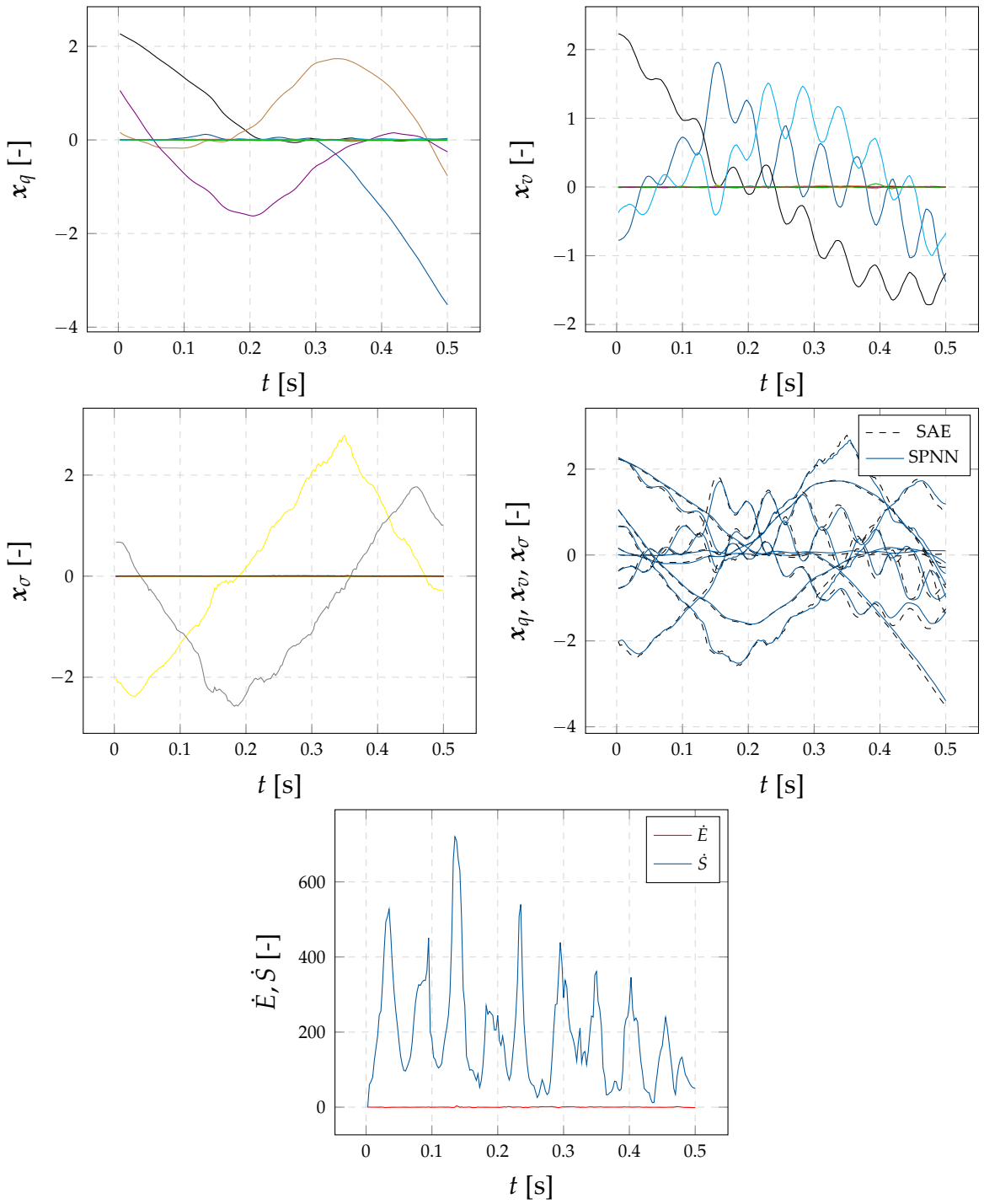


Figure 5.5: Top and middle left: Time evolution of the latent variables encoded with the sparse autoencoder (SAE) in the hyperelastic rolling tire problem. The bottleneck has $N_d = 40$ neurons and the learning algorithm sparsifies them to a dimensionality of $d = 9$ relevant latent variables. Middle right: Time evolution of the relevant latent variables integrated in time by the structure-preserving neural network (SPNN). Bottom: Evolution of the time derivative of the energy ($\dot{E}$) and entropy ($\dot{S}$) of the latent variables.

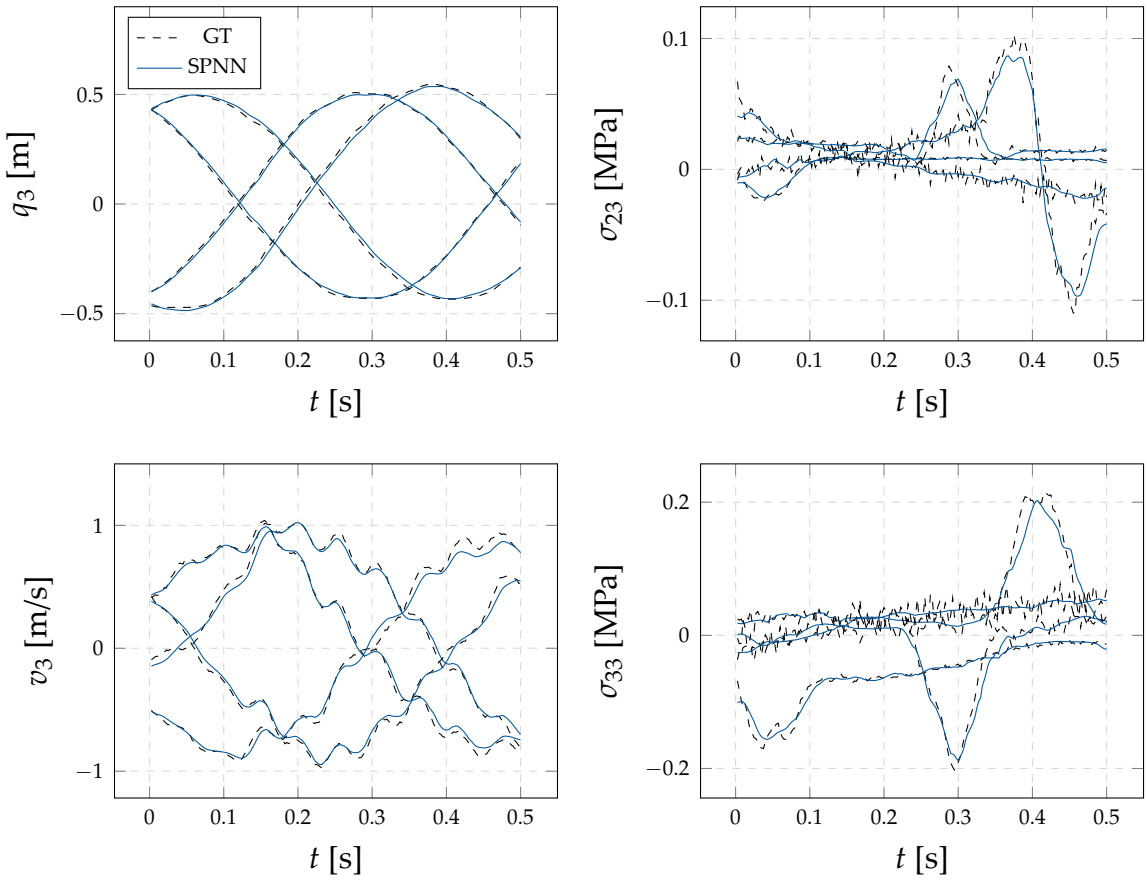


Figure 5.6: Results of the complete integration scheme (SPNN) with respect to the ground truth simulation (GT) for 4 different nodes and 4 different variables (q_3, v_3, σ_{33} and σ_{23}) of the hyperelastic rolling tire database.

Table 5.4: Mean squared error of the SPNN integration scheme and the unconstrained (UC) approach for the 12 state variables of the rolling tire example, reported for the complete trajectory.

State variable (z_i)	MSE^{SPNN}	MSE^{UC}
q_1 [m]	$2.07 \cdot 10^{-4}$	$2.76 \cdot 10^{-1}$
q_2 [m]	$8.09 \cdot 10^{-7}$	$1.57 \cdot 10^{-5}$
q_3 [m]	$6.25 \cdot 10^{-5}$	$5.75 \cdot 10^{-2}$
v_1 [m/s]	$7.95 \cdot 10^{-3}$	4.26
v_2 [m/s]	$3.79 \cdot 10^{-5}$	$7.00 \cdot 10^{-4}$
v_3 [m/s]	$1.78 \cdot 10^{-2}$	4.39
σ_{11} [MPa]	$2.04 \cdot 10^{-4}$	$4.71 \cdot 10^{-3}$
σ_{22} [MPa]	$1.76 \cdot 10^{-4}$	$4.07 \cdot 10^{-3}$
σ_{33} [MPa]	$2.70 \cdot 10^{-4}$	$5.35 \cdot 10^{-3}$
σ_{12} [MPa]	$6.05 \cdot 10^{-5}$	$1.50 \cdot 10^{-3}$
σ_{13} [MPa]	$1.48 \cdot 10^{-4}$	$3.03 \cdot 10^{-3}$
σ_{23} [MPa]	$2.73 \cdot 10^{-5}$	$6.67 \cdot 10^{-4}$

5.5 Conclusions

In this work, we develop a technique to learn the latent dimensionality of a physical system from data and obtain a thermodynamics-aware time integrator, which guarantees the fulfillment of the laws of thermodynamics. This technique is applied to two different physical systems. The Couette flow in a viscoelastic fluid is reduced from $D = 400$ dimensions to $d = 4$ dimensions, whereas the rolling tire is reduced from $D = 49680$ dimensions to $d = 9$ dimensions. The physically informed integrator is then able to predict the full time evolution of the set of state variables with similar precision reported in previous work [97, 72].

If compared to previous works of the authors in the field, the use of autoencoders to unveil the dimensionality of the embedding manifold clearly outperforms the results obtained by classical (linear) model order reduction techniques, specially in highly nonlinear state variables such as the rolling tire velocity. In addition, it is worth highlighting the fact that the method is able to detect the true dimensionality of the data, with no need to call to different codes rely on additional methods, such as k-PCA or TDA (Topological Data Analysis), for instance for this purpose. The right thermodynamic setting also ensures the consistency and stability of the full-order dynamics, after projecting back the reduced-order results to the physical space, achieving better results than an unconstrained approach with no physical restrictions.

This method has similar disadvantages as the discussed model order reduction techniques. The input of the reduction phase has fixed dimensions meaning that is only valid for the original meshed domain. Another limitation is the autoencoder oversmoothing of high frequency features in the full-order dynamics, which is a well known problem in neural networks called *spectral bias* [213]. There are several ideas to overcome these limitation in Chapter 8.

Part IV

Applications to Complex Systems

Port-Metriplectic Neural Networks

6

In this chapter, we develop inductive biases for the machine learning of complex physical systems based on the port-Hamiltonian formalism. To satisfy by construction the principles of thermodynamics in the learned physics, conservation of energy and non-negative entropy production, we modify accordingly the port-Hamiltonian formalism so as to achieve a port-metriplectic one. We show that the constructed networks are able to learn the physics of complex systems by parts, thus alleviating the burden associated to the experimental characterization and posterior learning processes of this kind of systems. Predictions can be done, however, at the scale of the complete system. Examples are shown on the performance of the proposed technique. The content of this chapter is included in the following publication [96]:

Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
Port-metriplectic neural networks: thermodynamics-informed machine learning of complex physical systems
Computational Mechanics, 72, 553–561 (2023)

6.1 Introduction

As we have already seen in previous chapters, the possibility of developing *learned simulators* has attracted an important research activity in the computational mechanics community and beyond. By learned simulators we mean methodologies able to learn from data the dynamics of a physical system so as to perform accurate predictions about previously unseen situations without the burden associated to the construction of numerical models. Among their advantages we can cite that they are based on reusable architectures, can be optimized to work under really stringent real-time feedback rates, and are specially well suited for optimization and inverse problems.

However, in real engineering applications, physical systems are usually composed of smaller subsystems interacting with each other. There is a vast number of examples in which this is the case: similar components assembled into bigger mechanisms, fluid-structure interaction, multiphysics simulations, etc. Thus, it is interesting not only to study isolated closed systems, but also opened systems which might interact with their surroundings. In this case, the use of model order reduction might be difficult to achive, as the system might have very different complex geometries and governing equations so finding the reduction and inverse mapping is cumbersome.

In this chapter we develop a novel strategy based on the port-Hamiltonian formalism used to model open conservative systems [256, 15, 217], which we extend so as to comply with the first and second principles of thermodynamics by construction. Based on this new formalism, which we call port-metriplectic, we construct a deep neural network methodology to learn the physics of complex systems from data. The resulting port-metriplectic networks will comply by construction with the principles of thermodynamics—that can be enforced through hard or soft constraints—while they allow to analyse complex systems by parts. These parts will then communicate through energy ports to construct the final complex system.

6.2 Methodology

As presented in Section 2.3, many works have used the Hamiltonian principles of reversible dynamics as an inductive bias to learn conservative systems. We have justified that relevant real world phenomena include dissipative effects which do not guarantee energy conservation.

This thesis solves that problem by using a metriplectic model which accounts for both reversible and irreversible dynamics by including a second dissipative potential function, considering the whole as a closed system. However, the same problem can be seen as a conservative Hamiltonian system perturbed with an external force contribution, i.e. an open system interacting with its surroundings. This is the idea of port-Hamiltonian models, mainly developed in the control theory field.

6.2.1 Port-Hamiltonian Neural Networks

For dissipative systems, the easiest form of the evolution Eq. (3.1) could be, perhaps, a *gradient flow* [104]. Their evolution can be established after some (dissipation) potential $\mathcal{R}$ in the form [49]

$$\dot{z} = -\frac{\partial \mathcal{R}}{\partial z}.$$

Recently, the so-called *Symplectic ODE nets* (symODEN) [286, 285], have tackled the issue of introducing dissipation in the learned description of the physics. It is also

the approach followed in [82]. More recently, two distinct works have tackled the dissipation problem by relaxing equivariance in the networks [89, 265].

These works seem to be closely related to the vast corps of literature on the *port-Hamiltonian* approach to dynamical systems [256, 217, 15]. Port-Hamiltonian systems assume an evolution of the system in the form

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix} - \boldsymbol{D}(\boldsymbol{q}) \right) \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial \boldsymbol{q}} \\ \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{g}(\boldsymbol{q}) \end{bmatrix} \boldsymbol{u}, \tag{6.1}$$

where $(\boldsymbol{q}, \boldsymbol{p})$ are the generalized position and momenta, dissipation is included by adding a symmetric and positive semi-definite matrix $\boldsymbol{D}$, and control is considered through an actuation term $\boldsymbol{u}$ and a non-linear function of the position $\boldsymbol{g}(\boldsymbol{q})$. Eq. (6.1) reduces to the Hamiltonian description if no dissipation nor control are considered. Here, we have assumed a canonical form for the Hamiltonian, i.e., that it depends on a set of variables $\boldsymbol{z} = \{\boldsymbol{q}, \boldsymbol{p}\}$. More general forms can be expressed similarly.

The true advantage of using port-Hamiltonian formalisms as inductive biases in the learning procedure stems from the fact that, on one side, they allow the introduction of dissipation and control and, on the other, they model open systems (as opposed to classical Hamiltonian descriptions where energy conservation assumes inherently that the system is closed) [50].

Therefore, the use of port-Hamiltonian formalisms as inductive biases in learning processes is extremely interesting. However, classical port-Hamiltonian schemes do not guarantee a priori to comply with the laws of thermodynamics, see [42]. Since entropy is not explicitly defined in the classical port-Hamiltonian formulation, it is difficult to impose the fulfillment of the second principle of thermodynamics. Therefore, we suggest to extend the GENERIC formalism to open systems so as to develop alternative port-metriplectic biases.

6.2.2 Port-Metriplectic Neural Networks

Very few works exist, to the best of our knowledge, on the development of GENERIC formulations for open systems, that may lead to the development of *port-metriplectic* formulations. Maybe the only exception is [198], later on revisited by [11, 19], both published in conference proceedings and, of course, with no machine learning approximations. Both approaches are essentially identical, and start from the bracket formulation of GENERIC in Eq. (2.9). For open systems, these brackets take the form

$$\{\cdot,\cdot\} = \{\cdot,\cdot\}_{\text{bulk}} + \{\cdot,\cdot\}_{\text{boun}},$$

and

$$[\cdot,\cdot] = [\cdot,\cdot]_{\text{bulk}} + [\cdot,\cdot]_{\text{boun}}.$$

In other words, both brackets are decomposed additively into bulk and boundary contributions. With this decomposition in mind, the GENERIC principle now reads

$$\dot{z} = \{z, E\}_{\text{bulk}} + [z, S]_{\text{bulk}} = \{z, E\} + [z, S] - \{z, E\}_{\text{boun}} - [z, S]_{\text{boun}}. \tag{6.2}$$

The degeneracy conditions and must be satisfied by the bulk operators only, since it is possible, in general, that there may be a reversible flux of entropy at the boundary or, equivalently, an irreversible flux of energy at the boundary [198],

$$\boldsymbol{L}_{\text{bulk}}(z)\frac{\partial S_{\text{bulk}}}{\partial z} = \mathbf{0},$$

and

$$\boldsymbol{M}_{\text{bulk}}(z)\frac{\partial E_{\text{bulk}}}{\partial z} = \mathbf{0},$$

The particular form of the boundary terms in Eq. (6.2) depends, of course, of the particular phenomenon under scrutiny, but in a general way it can be expressed using $\boldsymbol{L}$ and $\boldsymbol{M}$ operators as

$$\dot{z} = \boldsymbol{L}\frac{\partial E}{\partial z} + \boldsymbol{M}\frac{\partial S}{\partial z} - \boldsymbol{L}_{\text{boun}}\frac{\partial E_{\text{boun}}}{\partial z} - \boldsymbol{M}_{\text{boun}}\frac{\partial S_{\text{boun}}}{\partial z}. \tag{6.3}$$

More particular expressions can be developed if we know in advance some properties of the system at hand. For instance, in the first numerical experiment we deal with a double pendulum by learning the behaviour of each pendulum separately. If we know in advance that the only boundary term comes from the energy-entropy pair transmitted by the other pendulum, and no other external contribution is present, more detailed assumptions in the form of degeneracy conditions can be assumed. This may lead to a decrease in learning time or the employ of less data.

Fig. 6.1 sketches the approach developed herein for complex systems. In the next section we explore the particular form that these terms could acquire for both finite and infinite dimensional problems.

We propose two learning procedures which correspond to different level of information available of the dynamics of the system. In the first example, we focus on two coupled subsystems in which we learn the self and boundary contributions of both subsystems to the global dynamics of the problem. This is the case when the interest is focused on the complete system divided into smaller subsystems. In the second example, we suppose that the external influence is determined by a load vector as a result of an unknown external interaction with another subsystem. Thus, the learning procedure is focused on the self and boundary contributions of only one subsystem based on an external interaction. This case is convenient for applications where only partial information of the system is available.

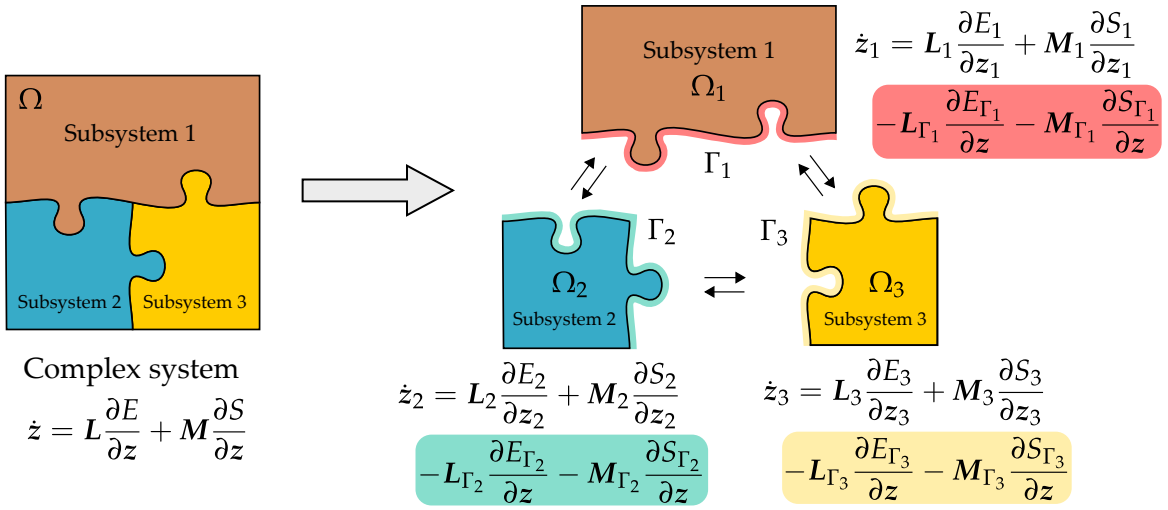


Figure 6.1: Complex system created as a connected group of subsystems Ω_i. The dynamics of any subsystem is described using a GENERIC formulation including conservative and dissipative terms, taking also into account the external contributions in the boundary terms Γ_i. Subsystems communicate between each other through ports by the exchange of energy and entropy.

6.3 Experiments

6.3.1 Double Thermoelastic Pendulum

Description

The first example is the double thermoelastic pendulum already presented in Section 3.4.1, see Fig. 3.2. The set of variables describing each individual pendulum are here chosen to be

$$\mathcal{S} = \{z = (q, p, s) \in (\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R})\}.$$

where q, p and s are the position, linear momentum and entropy of the pendulum mass. The evolution of the state variables of the second pendulum is defined as

$$\dot{z}_2 = L_2\frac{\partial E_2}{\partial z_2} + M_2\frac{\partial S_2}{\partial z_2} - M_{\text{boun},2}\frac{\partial S_{\text{boun},2}}{\partial z_2},$$

where the first two positive terms describe the self contribution of the simple pendulum (conservative and dissipative effects) and the third term describes the dissipative effect produced by the first pendulum affecting over the second pendulum.

On the other hand, the evolution of the state variables of the first pendulum is defined as

$$\dot{z}_1 = L_1\frac{\partial E_1}{\partial z_1} + M_1\frac{\partial S_1}{\partial z_1} - L_{\text{boun},1}\frac{\partial E_{\text{boun},1}}{\partial z_1} - M_{\text{boun},1}\frac{\partial S_{\text{boun},1}}{\partial z_1}, \tag{6.4}$$

where in this case the first two positive terms describe the self contribution of the

first simple pendulum (conservative and dissipative effects) and the third and fourth terms describe the external contribution on the conservative and dissipative parts, both produced by the influence of the second pendulum over the first pendulum.

Note that the first pendulum has no conservative contribution to the second pendulum, i.e., the term

$$\boldsymbol{L}_{\text{boun},2}\frac{\partial E_{\text{boun},2}}{\partial \boldsymbol{z}_2},$$

does not exist. However, there is a conservative contribution from the second pendulum on the first pendulum, see [222].

It is worth noting, as previously pointed out in [50], that the fact that every term in Eq. (6.4) depends on the state variables $\boldsymbol{z}_1$ makes the learning procedure more intricate, caused by its non-separable structure. This problem is not present if the port terms depend only on time, as it is the case in the next experiment. To overcome this limitation, we employ a structure-preserving neural network for each of the terms in Eq. (6.4). These networks share the weights, however, for both pendula, if they are known in advance to be identical.

The fact of using individual approximations of the dynamics of each subsystem (each pendulum) allows to use artificial neural networks of considerably smaller size with respect to an analysis of the whole problem using a larger number of variables to describe the global state as in Chapter 3.

Database and Hyperparameters

The database consists of 50 different simulations with random initial conditions of position $\boldsymbol{q}$ and linear momentum $\boldsymbol{p}$ of both masses m_1 and m_2 around a mean position and linear momentum of $\boldsymbol{q}_1 = [4.5,\ 4.5]$ m, $\boldsymbol{p}_1 = [2,\ 4.5]$ kg·m/s, and $\boldsymbol{q}_2 = [-0.5,\ 1.5]$ m, $\boldsymbol{p}_2 = [1.4,\ -0.2]$ kg·m/s respectively. The masses of the double pendulum are set to $m_1 = 1$ kg and $m_2 = 2$ kg, joint with springs of a natural length of $\lambda_1^0 = 2$ m and $\lambda_2^0 = 1$ m and thermal constant of $C_1 = 0.02$ J and $C_2 = 0.2$ J and conductivity constant of $\kappa = 0.5$. Note that the double pendulum constitutes a closed system as a whole, but this is not the case for each one of the simple pendula. Both start from a temperature of 300K. The simulation time of the movement is $T = 60$ s in $N_T = 200$ time increments of $\Delta t = 0.3$ s.

Results

The boxplot in Fig. 6.2 shows the statistics of the L2 relative error of the rollout train and test simulations.

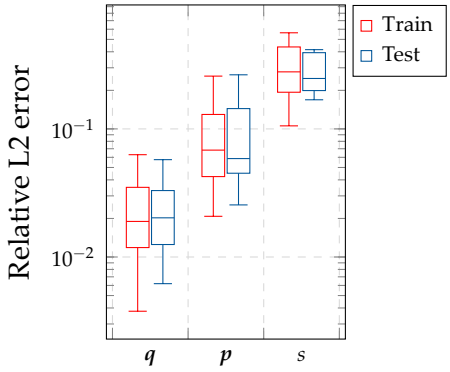


Figure 6.2: Box plots of the relative L2 error of the double pendulum. The state variables represented are position (q), momentum (p), and entropy (s).

6.3.2 Interacting Beams

Description

In this example we consider two viscoeleastic beams that can interact through contact between them, see Fig. 4.3, and whose physics are to be learned. Each individual viscoelastic beam is defined with the same geometry and material parameters as in Section 4.3.2. We assume that the necessary state variables for for a proper description of the beams are the position $\boldsymbol{q}$, its velocity $\boldsymbol{v}$ and the stress tensor $\boldsymbol{\sigma}$,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\sigma}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^6\},$$

at each node of the discretization of the beams. Since both beams are identical, we characterize only one of them and develop a port-metriplectic learned simulator for the joint system.

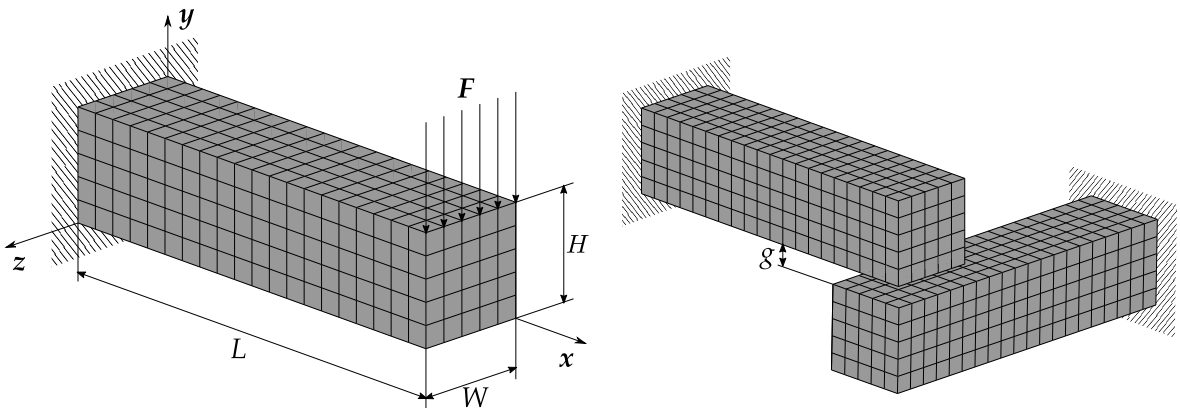


Figure 6.3: Interacting beams example. One single beam problem is analysed from simulation data. The resulting system is composed of two of these beams interacting together.

We use a thermodynamics-informed graph neural network developed in Chapter 4 to learn the self contribution of the dynamics, i.e. the first two terms of Eq. (6.3). The

boundary terms are learned using a standard structure-preserving neural network [97] with the additional input of the external forces applied to the beam.

Database and Hyperparameters

The dimensions of the beams are $H = 10$, $W = 10$ and $L = 40$. The finite element mesh from which data are obtained consisted of $N_e = 500$ hexahedral linear brick elements and $N = 756$ nodes. The constitutive parameters are $C_{10} = 1.5 \cdot 10^5$, $C_{01} = 5 \cdot 10^3$, $D_1 = 10^{-7}$ and $\bar{g}_1 = 0.3$, $\bar{g}_2 = 0.49$, $\tau_1 = 0.2$, $\tau_2 = 0.5$ respectively. A distributed load of $F = 10^5$ is applied in 52 different positions with an orientation perpendicular to the solid surface. Simulations were quasi-static and included $N_T = 20$ time increments of $\Delta t = 5 \cdot 10^{-2}$ s. Two identical beams are assembled in 90° with a gap of $g = 10$, as depicted in Fig. 4.3.

Results

The results are presented in Fig. 6.4. The error magnitude is similar as the reported in Chapter 4 in addition to the consistent formulation of port-metriplectic dynamics.

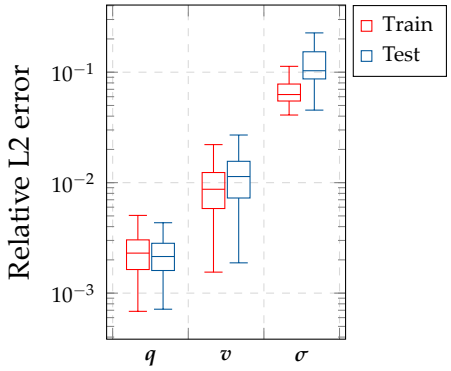


Figure 6.4: Box plots of the relative L2 error of the interacting beams example. The state variables represented are position (q), velocity (v), and stress tensor (σ).

6.4 Conclusions

In this chapter we have made two main contributions. On one side, the development of port-Hamiltonian-like approximations for dissipative open systems that communicate with other systems by exchanging energy and entropy through ports in their boundaries. This formulation extends the classical port-Hamiltonian approaches while guaranteeing the fulfillment of the laws of thermodynamics (conservation of energy in the bulk system, non-negative entropy production). The resulting formulation, which we refer to as port-metriplectic, presents a rigorous thermodynamic description of the dissipative behaviour of the system.

On the other hand, the just developed formulation is employed as an inductive bias for the machine learning of the physics of complex systems from measured data. This bias is developed as a soft constraint in the loss term, although it can also be imposed straightforwardly as a hard constraint.

The resulting neural networks, for which we have formulated two distinct versions, one based on standard multilayer perceptrons, and a second one based on graph neural networks, have shown an excellent performance. Error bars are equivalent to those obtained in previous works of the authors, by employing a closed-system approach to the same physics. The new approach opens the door to the development of learned simulators for complex systems through piece-wise learning of the physical behaviour of each of its components. The final, global simulator is then obtained by assembling each piece through their ports.

Real-time Deep Simulators for Augmented Reality

7

The imminent impact of immersive technologies in society urges for active research in real-time and interactive physics simulation for virtual worlds to be realistic. In this context, realistic means to be compliant to the laws of physics. In this chapter we present a method for computing the dynamic response of (possibly non-linear and dissipative) deformable objects induced by real-time user interactions in mixed reality using deep learning. The graph-based architecture of the method ensures the thermodynamic consistency of the predictions, whereas the visualization pipeline allows a natural and realistic user experience. Two examples of virtual solids interacting with virtual or physical solids in mixed reality scenarios are provided to prove the performance of the method. The content of this chapter is included in the following publication [99]:

Q. Hernández, A. Badías, F. Chinesta, & E. Cueto
Thermodynamics-informed neural networks for physically realistic mixed reality
Computer Methods in Applied Mechanics and Engineering, 407, 115912 (2023)

7.1 Introduction

Computer science advances in the last decades led us to experience in a relatively short lapse of time three major technological innovations: the personal computer, the Internet, and mobile devices. Currently, we are at the beginning of a *fourth paradigm* of computing innovations involving immersive technologies such as Virtual Reality (VR), Augmented Reality (AR) or Mixed Reality (MR). All this is possible due to huge advances in machine learning techniques and hardware improvements applied to computer graphics and computer vision.

It is clear that this new paradigm seeks to revolutionize technology in the next years and will have a great impact in society such as smart cities [3, 258], new teaching methods [224] or economic paradigms [261]. Technology companies have already

created numerous digital platforms, such as the Metaverse [190, 136] or the Omniverse [111, 148], in order to develop their own immersive technologies.

In most of the cases, virtual worlds are required to resemble our real world as much as possible, so many disciplines come into play (traditionally, computer graphics and computer vision). However, virtual worlds need to be dynamic rather than static so the user can responsively interact with a changing virtual world. From that perspective, physics simulation plays a major role and it is required to be real-time. On the one hand, current real-time physic engines rely on severe simplifications of the governing dynamical equations and are limited to very simple material models and constitutive phenomena. On the other hand, classical engineering methods for solid and fluid simulations, such as the finite element or finite differences methods, have the consistency of decades of theoretical research in terms of the convergence to a consistent physical solution but are too expensive to achieve real-time framerates. However, these last methods can be used to generate a rich and consistent database to train a fast AI accelerated by the recent advances in machine learning procedures.

Real-time physics engines such as PhysX [167, 46] or Havok [264] have mostly been developed in the videogame industry. Even if those engines allow to program custom dynamical models, they usually rely on simplified mass-spring models or rigid body dynamics to achieve high framerates in modern videogames. Multiple research lines remain open trying to leverage the physical consistency of the results with low computational requirements.

Several authors solved the mentioned problems by creating a reduced order model of the system [62, 30], as already discussed in Chapter 5. However they all have similar disadvantages: as the solution is already precomputed, they are unable to handle different mesh discretizations and fail to generalize to unseen configurations. Other approaches are based on standard multilayer perceptrons, used as a collocation method for residual minimization [61, 196] or specific formulations for contact mechanics [223, 219]. However, these are engineered ad hoc for special dynamical systems and also require the prior knowledge of the governing equations.

In this work, we aim to merge the physical consistency of classical simulation techniques with the speed of real-time physics engines using a deep learning approach to develop a real-time interactive simulator. Although the formulation is general for a wide variety of dynamical systems, we focus on nonlinear solid mechanics. The results are consistent with the laws of thermodynamics by construction and are able to achieve real-time performance in general load cases which were not previously seen by the network.

7.2 Problem Statement

The present chapter focuses on the deformation of virtual solid objects. Thus, we use the dynamical equilibrium equation of non-linear solid mechanics which balances the external and internal body forces with the acceleration of the solid. This is,

$$\nabla \cdot \boldsymbol{P} + \boldsymbol{B} = \rho \ddot{\boldsymbol{u}} \text{ in } \Omega_0, \tag{7.1}$$

where $\boldsymbol{B}$ represents the volumetric force applied to the body and $\boldsymbol{P}$ the first Piola-Kirchhoff stress tensor. $\Omega_0 = \Omega(t = 0)$ represents the undeformed configuration of the virtual solid. The solution is subjected to appropriate boundary conditions

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{X}) &= \bar{\boldsymbol{u}} \text{ on } \Gamma_u, \\ \boldsymbol{P}\boldsymbol{N} &= \bar{\boldsymbol{t}} \text{ on } \Gamma_t, \end{aligned}$$

with Γ_u and Γ_t representing the essential (Dirichlet) and natural (Neumann) portions of the boundary $\Gamma = \partial\Omega$ of the solid. $\boldsymbol{X}$ is the undeformed position, $\boldsymbol{N}$ is the unit vector normal to $\Gamma = \partial\Omega_0$ and $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{t}}$ are the applied displacement and traction respectively. To complete the problem, some relationship between kinematic variables (displacements, strain) and dynamic variables (stresses) must be assumed. The constitutive equation is here chosen to be hyperelastic, with a strain energy function per unit volume Ψ defined such that

$$\boldsymbol{S} = \frac{\partial \Psi}{\partial \boldsymbol{E}},$$

where $\boldsymbol{S}$ is the second Piola-Kirchhoff and $\boldsymbol{E}$ is the Green-Lagrange strain tensor. Viscoelastic effects are also considered using variable shear relaxation modulus via Prony series. The objective of the method is to solve Eq. (7.1) in a real-time interactive interface with a physics-based neural network trained with high fidelity solutions.

7.3 Methodology

7.3.1 Integration Method

We use the graph-based integration scheme from Chapter 4 adapted to the augmented reality setup. Its objective is to learn not the outcome of a given simulation under different conditions such as forces or boundary conditions, but the actual *physics* taking place, such that the learned simulator is not sensitive to changes in the mesh, for instance. A scheme of the algorithm is presented in Fig. 7.1.

We assume our virtual solids to be viscous-hyperelastic, so that the state variables for the proper description of their evolution in terms of the GENERIC formalism are

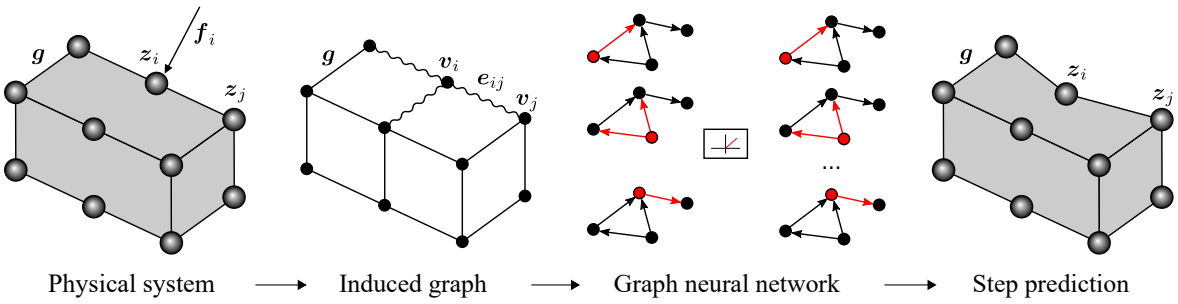


Figure 7.1: Thermodynamics-informed graph neural network architecture. The system is described as a set of state variables z_i, global simulation parameters g and external boundary conditions f_i. A graph $\mathcal{G}$ is constructed from the information of the physical system, defining vertex features v_i, edge features e_{ij} and global features g. The graph features are processed with a message-passing graph neural network. The step prediction is performed using the GENERIC integration scheme, which is repeated iteratively to get the complete rollout of the simulation.

the position q, velocity v and the stress tensor σ,

$$\mathcal{S} = \{z = (q, v, \sigma) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^6\}.$$

The edge feature vector contains the relative deformed position between nodes, to give a distance-based attentional flavour to the graph processing blocks and translational invariance. The velocity and stress tensor components are part of the node feature vector, concatenated to a two-dimensional one-hot vector n which represent the encastre and solid nodes respectively. The external load vector f_i is included in the node processor MLP as an external interaction. No global feature vector is needed in this case, resulting in the following feature vectors:

$$e_{ij} = (q_i - q_j, \|q_{ij}\|_2), \quad v_i = (v, \sigma, n).$$

Thus, the dimensions of the feature vectors are $F_e = 4$, $F_v = 11$, $F_f = 3$ and $F_g = 0$.

7.3.2 Vision System

For an augmented or mixed reality application, we need to include virtual objects in a real scene. For that, it is necessary to have a sensor able to get information about the physical environment and a screen device to plot the resulting image. In the present work, we use a ZED Mini stereo vision system from Stereolabs, which is able to retrieve both a depth and RGB image of the captured snapshot. We plot the resulting real-time video stream in a computer screen, but could be extended to a VR headset or AR glasses.

As the laws of motion are independent with respect to any frame of reference (objectivity), we use the model space to compute the dynamical state variables of the

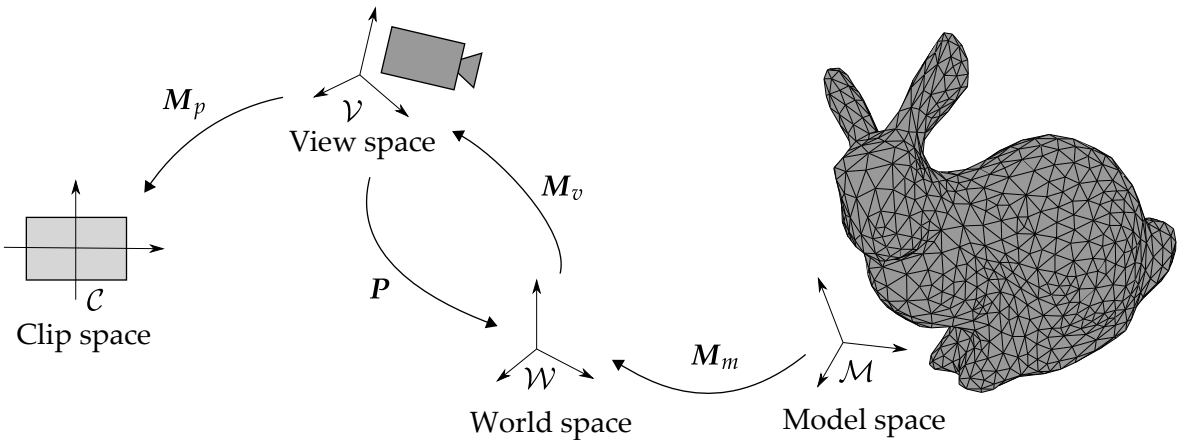


Figure 7.2: Model-view-projection transformation between the model and the clip coordinates. The dynamics and interactions are computed in the model space as 3D coordinates, whereas the visualization pipeline requires a 2D clipping coordinates in a fixed range between -1 and 1.

system. However, the visualization pipeline requires the coordinates to be in a normalized range of $\mathcal{C} = [-1,1] \times [-1,1]$ called the clip space. The transformation between the 3D model space and the 2D clip space is achieved by the definition of the *Model-View-Projection matrix* (see Fig. 7.2). From now, the point coordinates are supposed to be in homogeneous coordinates.

- **Model**: The dynamics are computed in a local frame of reference called the model space $\mathcal{M}$. The position, orientation and scale of the model can be defined as a set of transformation matrices $\boldsymbol{T}_m$, $\boldsymbol{R}_m$ and $\boldsymbol{s}_m$ respectively with respect to the world space $\mathcal{W}$, considered to be the usual Euclidean space $\mathbb{R}^3$.

$$\boldsymbol{x}_{\text{world}} = \boldsymbol{M}_m \boldsymbol{x}_{\text{model}} = \boldsymbol{T}_m \boldsymbol{R}_m \boldsymbol{s}_m \boldsymbol{x}_{\text{model}}.$$

- **View**: In a similar fashion, the viewing camera also has a model matrix defining its position in the world space, which is commonly designed as the extrinsic parameters $\boldsymbol{P}_v$ of the camera pose. This information can be obtained using a monocular pose estimator or triangulating with a stereo vision system. Thus, the camera or view space $\mathcal{V}$ can be determined by a set of transformations from the world space using the camera extrinsic parameters given by a rotation $\boldsymbol{R}_v$ and translation $\boldsymbol{T}_v$ matrices.

$$\boldsymbol{x}_{\text{view}} = \boldsymbol{M}_v \boldsymbol{x}_{\text{world}} = {\boldsymbol{P}_v}^{-1} \boldsymbol{x}_{\text{world}} = (\boldsymbol{T}_v \boldsymbol{R}_v)^{-1} \boldsymbol{x}_{\text{world}}.$$

- **Projection**: The last transformation is in charge of projecting the 3D coordinates into the 2D clip space $\mathcal{C}$. First, in order to get a realistic visualization, we

use a perspective projection $\boldsymbol{M}_p$ based on the camera viewing frustrum

$$\boldsymbol{M}_p = \begin{pmatrix} \cot\frac{\alpha}{2} & 0 & 0 & 0 \\ 0 & \cot\frac{\alpha}{2} & 0 & 0 \\ 0 & 0 & -\frac{z_{\text{far}}+z_{\text{near}}}{z_{\text{far}}-z_{\text{near}}} & -\frac{2z_{\text{far}}z_{\text{near}}}{z_{\text{far}}-z_{\text{near}}} \\ 0 & 0 & -1 & 0 \end{pmatrix},$$

where α is the field of view of the camera and z_{near} and z_{far} are the minimum and maximum distances of the clipping plane. This creates a normalized 3D viewing box of the camera, which is then projected in the 2D clip space by the vertex shader. Thus, the final coordinates can be computed using the following equation:

$$\boldsymbol{x}_{\text{clip}} = \boldsymbol{M}_p\boldsymbol{x}_{\text{view}} = \boldsymbol{M}_p\boldsymbol{M}_v\boldsymbol{M}_m\boldsymbol{x}_{\text{model}}. \tag{7.2}$$

By defining the Model-View-Projection matrix as $\boldsymbol{M}_p\boldsymbol{M}_v\boldsymbol{M}_m$ we can compute the clip coordinates directly from the model coordinates computed by the thermodynamics-informed neural network.

7.3.3 Visualization System

The visualization of the resulting image is performed using OpenGL and the GLU library. This pipeline requires the definition of two spatial functions, run sequentially on the GPU, called the vertex and fragment shaders. The vertex shader defines the vertex position of the entities to display whereas the fragment (or texture) shader define the RGBA colors for each rasterized pixel.

Each vertex position is computed using Eq. (7.2) from the deformed configuration coordinates and the polygons are drawn based on the connectivity matrix of each solid. The color of each vertex is computed directly from the neural network prediction using a fixed colormap. Additionally, a basic lighting model was added to the fragment shader to increase the realism of the virtual objects using the *Phong shading*. This model computes each RGB pixel intensity as

$$\boldsymbol{I} = k_a\boldsymbol{i}_a + k_d(\boldsymbol{l}\cdot\boldsymbol{n})\boldsymbol{i}_d + k_s(\boldsymbol{r}\cdot\boldsymbol{v})^\beta\boldsymbol{i}_s,$$

where k_a, k_d and k_s are the ambient, diffuse and specular reflection constants, β is the shininess constant of the material and $\boldsymbol{i}_a$, $\boldsymbol{i}_d$ and $\boldsymbol{i}_s$ are the RGB color intensities of the ambient, diffuse and specular components. The illumination varies depending on the geometry of the scene and camera position, where $\boldsymbol{n}$ is the surface normal vector, $\boldsymbol{l}$ and $\boldsymbol{r}$ are the directions of the light source and its perfect reflection, $\boldsymbol{v}$ is the viewing direction and "$\cdot$" is the dot product.

We have also implemented a depth or z-buffer in the fragment shader which compares on each pixel the depth of virtual and real objects on the scene and renders

only the closest to the camera. This accounts for every occlusion that the vision system may encounter. The depth of the scene seen by the camera is computed directly using the stereo vision system.

7.3.4 Collision and Contact

We use a high-fidelity hand tracker from MediaPipe [278] which provides an accurate localization of the finger tips. By using the inverse transformation of Eq. (7.2), we can compute the 3D coordinates of the finger tips in the model space and compute the distance to each node of the virtual object. When this distance is less than a small threshold, collision is detected and a prescribed force is applied to the model. A similar procedure is applied to the collision of several virtual objects.

The code is implemented in Python using the PyOpenGL wrapper for the visualization and Pytorch Geometric [58] for the deep learning training and evaluation. The videos are generated using a standard desktop computer with a single Nvidia RTX2070 GPU.

7.4 Experiments

7.4.1 Bending Beams

Description

The first system is composed of two interacting viscoelastic beams as described in Chapter 6. Both identical beams are assembled with a small gap between each other, allowing for a contact interaction, as depicted in Fig. 6.3.

Database and Hyperparameters

The dimensions of the beams are $H = 10$, $W = 10$ and $L = 40$. The finite element mesh from which data are obtained consists of $N_e = 500$ hexahedral linear brick elements and $N = 756$ nodes. The constitutive parameters of the hyperelastic strain energy potential are $C_{10} = 1.5 \cdot 10^5$, $C_{01} = 5 \cdot 10^3$, $D_1 = 10^{-7}$ and $\bar{g}_1 = 0.3$, $\bar{g}_2 = 0.49$, $\tau_1 = 0.2$, $\tau_2 = 0.5$ respectively for the two-term Prony series. A distributed load of $F = 10^5$ is applied in 52 different positions with a perpendicular direction to the solid surface. Each simulation is composed of $N_T = 20$ time increments of $\Delta t = 5 \cdot 10^{-2}$ s. Both beams are assembled in 90° with a gap of $g = 10$.

The graph neural network vertex and edge MLPs have two layers of $F_h = 100$ neurons each, with 10 message passing sequential blocks. The training was performed for $N_{\text{epoch}} = 1800$ epochs and learning rate $l_r = 10^{-4}$.

Results

Fig. 7.3 shows a real-time video sequence generated using the presented algorithm. The interaction of both the real objects (finger tips) and the virtual objects are simulated smoothly at more than 30 frames per second. It is important to highlight that during a video sequence the trained neural network is able to generalize to previously unseen configurations. The quantitative errors by the neural network in the real-time rollout predictions are shown in Fig. 7.6a, which remain below 1% in position and velocity and 10% in the stress tensor field.

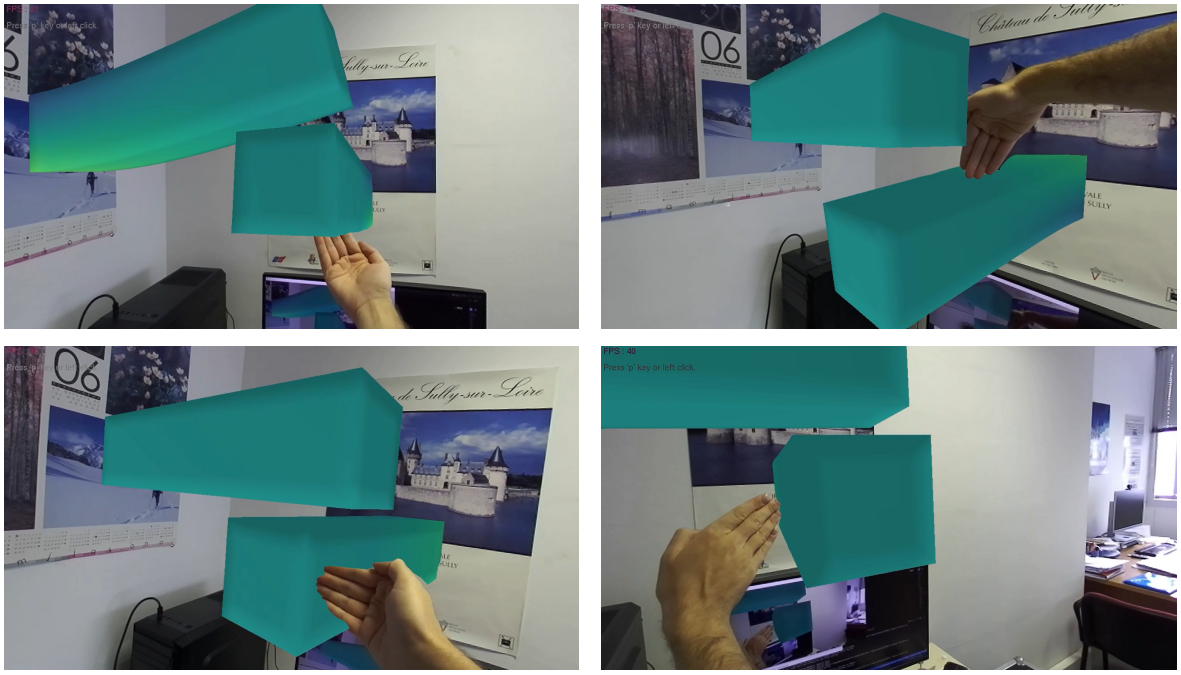


Figure 7.3: Frames extracted from the interacting beams sequence. Color encodes the x-x stress field component associated with the displacement imposed by contact with a real object.

The computational cost of the high fidelity FEM simulations used for the training of the neural network is 15 s per simulation, up to 795 s for the whole dataset, and 243 Mb in memory storage. Conversely, the training time of the neural network is 4.5 h and the mean evaluation time is 9 ms per snapshot with a memory storage of 12.3 Mb. Thus, the use of a deep learning approach allows a drastic reduction of the online computation time with the inconvenient of larger offline training time. It is also worth noting that the network parameters are more than 10 times lighter in terms of memory storage.

7.4.2 Stanford Bunny

Description

The second example is a bunny mesh from the Stanford 3D scanning repository, which is a more complex geometry as the one shown in the previous example. It is a standard reference model in computer graphics research. The model is represented in Fig. 7.4.

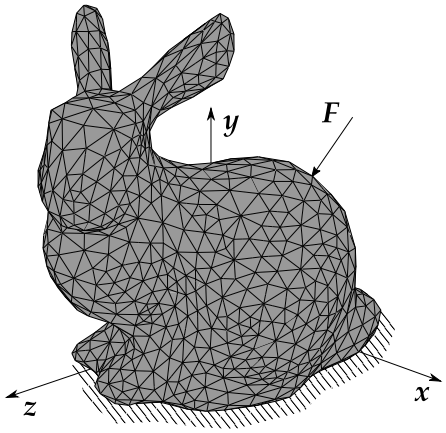


Figure 7.4: Bunny mesh geometry with boundary conditions. A concentrated force is applied to random nodes and the bottom of the mesh is encastred.

Database and Hyperparameters

The finite element mesh from which data are obtained consisted of $N_e = 4941$ tetrahedral linear elements and $N = 1352$ nodes. The constitutive parameters of the hyperelastic strain energy potential are $C_{10} = 2.6 \cdot 10^{-1}$ and $D_1 = 4.9 \cdot 10^{-2}$. Similarly as the previous case, a concentrated load of $F = 1$ is applied in 100 different positions with a perpendicular direction to the solid surface. The body is fixed to the ground plane by disabling displacements and rotations at the lower model nodes. Each simulation is composed of $N_T = 20$ time increments of $\Delta t = 5 \cdot 10^{-2}$ s.

The graph neural network vertex and edge MLPs have two layers of $F_h = 100$ neurons each, with 10 message passing sequential blocks. The training was performed for $N_{\text{epoch}} = 1800$ epochs and learning rate $l_r = 10^{-4}$.

Results

Fig. 7.5 shows a real-time video sequence generated using the bunny model, also with a minimum framerate of 30 frames per second. In this case, the stress field errors reported in Fig. 7.6b are higher due to the stress peaks at the single-node force application, which causes that the elasticity phenomena remain very local in space.

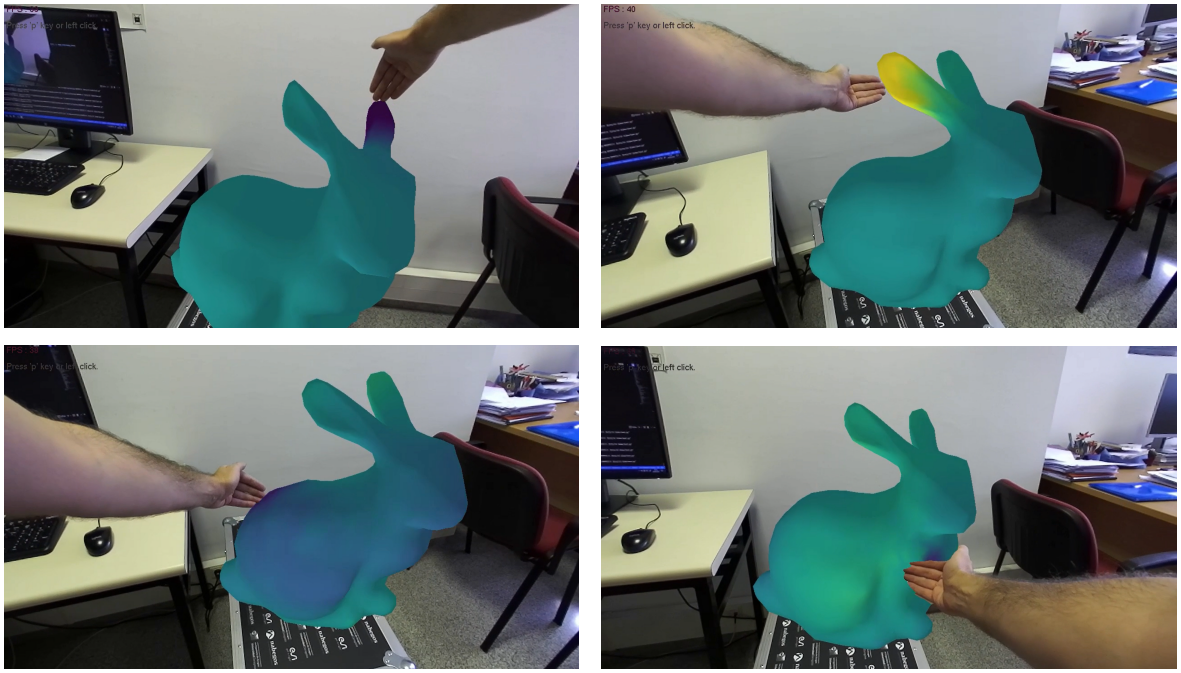
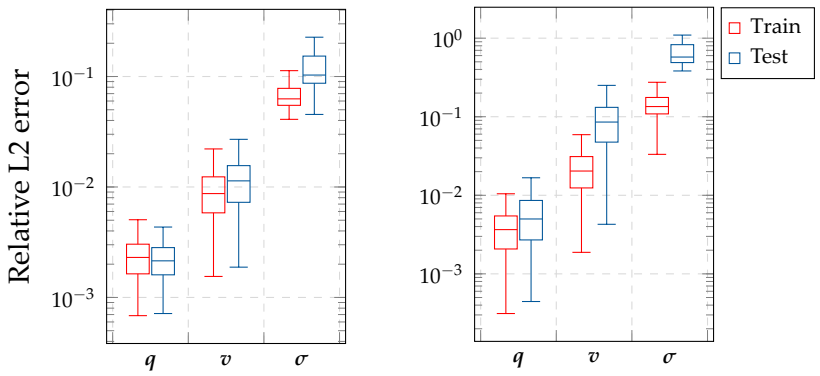


Figure 7.5: Frames extracted from the bunny sequence. Color encode the x displacement field component imposed by contact with a real object.

The computational cost of the high fidelity FEM simulations is 27 s per simulation, up to 2700 s for the whole dataset and 833 Mb in memory storage. The dataset and edge count is much larger in this example, which increases the training time of the neural network to 20 h and the evaluation time to 11 ms per snapshot with the same memory storage as the previous example. Thus, the data compression is even bigger in this case. To address the high training time and errors due to stress peaks, several future improvements are discussed in the next section.



(a) Beams (b) Bunny

Figure 7.6: Box plots of the relative L2 error of the bunny example. The state variables represented are position (q), velocity (v), and stress tensor (σ).

7.5 Conclusions

We presented a real-time augmented reality simulator, which enables a user to interact with virtual deformable solids. The predictions are computed using the GENERIC structure of the system learnt with a message passing graph neural network. The enforcement of such physics constraints guarantees the fulfilment of the first and second laws of thermodynamics. The resulting algorithm has a wide variety of applications not only in the entertainment industry, but also in engineering design or manufacturing, where the visualization of augmented data superimposed in a real or virtual object might redefine the next generation of industry 4.0 and digital twins.

Our method has several limitations which might be addressed as future work. The scalability to bigger meshes is a challenging task, as graph neural networks can suffer from *over-squashing* or bottlenecks [6, 250]. As the core physics engine used in this chapter is based on the TIGNNs, the limitations are very similar to Chapter 4. Additional methods as future work are discussed in Chapter 8.

The visualization and data acquisition are possible due to the graphics acceleration. Even if the current work was implemented in a desktop computer as a proof of concept, only a small fraction of all the computational resources were used, so it can be extended to AR/VR headsets or to modern mobile devices. For the same reason, higher framerates might be achieved by fine-tuning and optimizing the proposed network structure or porting the code to a higher-performance language such as C++. Occlusions in real-time augmented reality software still remain as an open problem in the computer vision community [203, 271]. The stereo depth estimation is an approximation which might cause image artifacts in singular camera poses. For instance, some works handle the problem by using deep learning [171] but it is out of the scope of this chapter.

This work is just a small step forward towards the new immersive technologies which can potentially deeply change our society. It is a multidisciplinary problem where computer vision, computer graphics, machine learning and computational mechanics must meet to define new algorithms for a new digital revolution.

Part V

Conclusions

8 Conclusions

This chapter provides an overview of the main results and contributions of the thesis divided by its three core parts: deep learning of dynamical systems, latent manifold learning and application to complex systems. In addition, future work is discussed based on the limitations of the presented techniques.

8.1 Concluding Remarks

Deep Learning of Dynamical Systems

This thesis analyses the use of state-of-the-art deep learning methods to learn simulators from data. For that purpose, we propose integration algorithms based on several inductive biases which improve generalization and the accuracy of the results:

- The *metriplectic bias* ensures the thermodynamical consistency of the results, i.e. energy conservation and entropy inequality. As opposed to many *black-box* approaches in scientific machine learning, we have shown that the structure-preserving techniques presented in this thesis have a certain level of interpretability in terms of energetic consistency of the results and mathematical properties of the integrator.
- The limitations of Chapter 3 are then solved in Chapter 4 with the use of *geometric biases*, which exploit the geometrical properties of the domain with the use of graph-based algorithms. This approach improves greatly the versatility of the algorithm, being able to generalize to unstructured domains and adapt to different meshes due to the permutation and translational invariances built upon it. The results are also competitive with state-of-the-art GNN physics simulator methods.

Both methods can be used if the governing equations are known or not. The metriplectic structure of dynamics can be enforced by knowing the corresponding Poisson

and dissipative operators, as in Chapter 3, or learnt based on data, as in Chapters 4 and 5.

Latent Manifold Learning

We have also proposed in Chapter 5 a reduced order model method to learn a sparse representation of high-dimensional systems. The main advantage of this technique is that the intrinsic dimensionality of the data is learnt by the encoding projection in an unsupervised manner even for highly nonlinear phenomena. Then, the structure-preserving methodologies can be applied in the more manageable reduced space. This technique drastically reduces the compute time of large systems, which is ideal for real-time applications with very stringent time limitations.

Application to Complex Systems

The last chapters of the thesis explore the viability of the developed methodologies to complex systems and real engineering applications.

- Chapter 6 show an extension of metriplectic systems to open domains where the energetic interactions with the environment are crucial for achieving the correct predictions. This study is relevant because real engineering systems are rarely computed as a whole but with its constituent components, in order to refine their design by different engineering teams or reduce the computational requirements. In that case, the individual models must be corrected with the energetic terms presented in this study.

  Additionally, this technique can be applied to systems which interact with a surrounding unknown system. In this case, the state vector of the external interaction is not known but only the perturbation produced to the main system. Both situations can be learnt by the port-metriplectic neural networks with similar accuracy to previous techniques.

- Chapter 7 integrates the proposed methods as a physics engine of an augmented reality application in deformable solids. This work shows the performance of thermodynamics-informed neural networks under real-time requirements. The demo tested in an structured and unstructured mesh show a great performance with stable framerates and smooth user interaction. Thus, the physically-sound integrators developed in this thesis could potentially be used in digital twin and industrial applications in which the physical consistency of the results is fundamental.

8.2 Future Work

This section proposes several research directions to overcome the limitations commented in the conclusion section of each individual chapter. These ideas emerged during the development of the thesis, considering that machine learning is a very active and productive research field.

In terms of interpretability, one could say that our models are a *grey-box* as they still use a trained neural network as the core computing unit. There is still room for improvement, by using the latests advances in symbolic regression [26, 255], which seek parsimonious closed models from data which might better approximate the objective function and are also faster to evaluate. It is specially interesting the use of symbolic regression in graph neural networks [41], because the node and edge blocks have a clear intuitive interpretation as node-wise and pair-wise interactions between particles. However, these techniques are limited if using normalization layers and latent vectors, which need not to be interpretable at all.

A higher order Runge-Kutta method could be introduced in the integration scheme in order to get higher solution accuracy [267]. However, it requires several forward passes through the neural net for each time step, incrementing the complexity of the integration scheme and the training process. Additionally, GENERIC-based integration schemes have showed very good performance even for first-order approaches [222]. A similar approach could be obtained with the use of RNNs or *Transformers* [257, 269]. This last architecture uses attention rather than recurrence to find temporal patterns in large contexts, and recently had increased popularity due to its successful use in Large Language Models (LLMs). This method is conceptually similar to a high-order integration scheme with variable weights based on the self-attention mechanism [29, 232].

A limitation of the present work is the use of synthetic instead of experimental data. A research field is opened to test the limits of the presented methodology applied to real captured data, and to study the influence of noise in the measurements of real-world applications, such as digital twins of industrial processes or real-time augmented/virtual reality environments. A more exhaustive analysis can be performed to evaluate the influence of noisy data to the integrator evolution, in order to add robustness to the method and even predict wider simulation times using incremental learning [151, 36].

As shown in Chapter 4, the message passing algorithm has several disadvantages when handling large domains which require fine grids. Future work may overcome these limitation by combining graph representations with model order reduction techniques, such as graph autoencoders [132] or U-net architectures [65, 275]. The idea is to replace deep message passing with various coarse-graining steps, allowing the boundary information to reach every node in the simulation domain while reducing the number of parameters of the neural network. This can also be mitigated

by implementing a global attention vector so that certain dynamical information is reached to all the nodes of the domain instantly.

Another interesting topic to extend our work is to improve the generalization and decrease the amount of training data via *equivariant* arquitectures [125, 233], which avoid data augmentation by exploiting the invariance to certain groups such as rotations $SO(3)$ or general Euclidean transformations $E(3)$.

In the case of the structure-preserving autoencoders in Chapter 5, the latent space of the reduction step can not only be sparsified, but also regularized using variational inference via Variational Autoencoders [129]. This Bayesian approach is convenient in cases where the latent variables are sampled or interpolated, and lead to smoother transitions between them. A future line of this work is to explore VAEs applied to physical systems and study their influence in the latent space topology and extrapolability. The use of graph-based architectures and implicit representations [31] can also overcome the limitation of the fixed input mesh format and improve generalization to varying meshes. To improve the over-smoothness of the reconstructed dynamics, a frequency-based encoding layer might be added to the input layers which transforms the dynamical features based on their frequency content and improves the learning procedure [246].

The use of graph neural networks in elasticity problems has numerous similarities to the peridynamic theory [239, 240]. This non-local formulation of solid mechanics has strong mathematical foundations and several advantages over discontinuities, such as fracture, and long-range force effects. It could be interesting to explore the adaptation of peridynamic concepts in the geometric deep learning context to learn more complex effects such as plasticity or fracture.

The techniques presented in this thesis have already been used in several real-world applications, such as the study of the sloshing effects of liquids in real-time [188] with promising results. There is a possible future research line towards augmented reality applications in a similar fashion of Chapter 7, regarding optimization of the proposed pipeline and extension to more complex geometries, load cases and material constitutive properties.

9 Conclusiones

En este capítulo se ofrece una visión general de los principales resultados y contribuciones de la tesis divididos en sus tres partes principales: aprendizaje profundo de sistemas dinámicos, aprendizaje de espacios latentes y aplicación a sistemas complejos. Además, se discute el trabajo futuro basado en las limitaciones de las técnicas presentadas.

9.1 Observaciones finales

Aprendizaje profundo de sistemas dinámicos

Esta tesis analiza el uso de métodos de aprendizaje profundo de última generación para aprender simuladores a partir de datos. Para ello, se proponen algoritmos de integración basados en varios sesgos inductivos que mejoran la generalización y la precisión de los resultados:

- El *sesgo metripléctico* asegura la consistencia termodinámica de los resultados, es decir, la conservación de la energía y la desigualdad de entropía. A diferencia de muchos enfoques *caja negra* en el aprendizaje automático científico, se ha demostrado que las técnicas de preservación de la estructura presentadas en esta tesis tienen un cierto nivel de interpretabilidad en términos de consistencia energética de los resultados y propiedades matemáticas del integrador.
- Las limitaciones del Capítulo 3 se resuelven posteriormente en el Capítulo 4 con el uso de *sesgos geométricos*, que explotan las propiedades geométricas del dominio con el uso de algoritmos basados en grafos. Este enfoque mejora enormemente la versatilidad del algoritmo, pudiendo generalizarse a dominios no estructurados y adaptarse a diferentes mallas gracias a las invarianzas de permutación y traslación construidas sobre él. Los resultados también son competitivos con los métodos de simulación física basados en grafos más avanzados.

Ambos métodos pueden utilizarse tanto si se conocen las ecuaciones del sistema como si no. La estructura metripléctica de la dinámica puede imponerse conociendo los correspondientes operadores de Poisson y disipativos, como en el Capítulo 3, o aprenderse basándose en datos, como en los Capítulos 4 y 5.

Aprendizaje de espacios latentes

También se ha propuesto en el Capítulo 5 un método de reducción de orden para aprender una representación reducida de sistemas de alta dimensionalidad. La principal ventaja de esta técnica es que la dimensionalidad intrínseca de los datos es aprendida por la proyección de codificación de forma no supervisada incluso para fenómenos altamente no lineales. A continuación, las metodologías de preservación de la estructura pueden aplicarse en un espacio reducido más manejable. Esta técnica reduce drásticamente el tiempo de cálculo de grandes sistemas, lo que resulta ideal para aplicaciones en tiempo real con limitaciones de tiempo muy estrictas.

Aplicación a sistemas complejos

Los últimos capítulos de la tesis exploran la viabilidad de las metodologías desarrolladas para sistemas complejos y aplicaciones reales de ingeniería.

- Capítulo 6 muestra una extensión de los sistemas metriplécticos a dominios abiertos donde las interacciones energéticas con el entorno son cruciales para obtener predicciones correctas. Este estudio es relevante porque los sistemas de ingeniería reales rara vez se calculan como un todo, sino mediante sus componentes constituyentes, con el fin de refinar su diseño por diferentes equipos de ingeniería o reducir los requerimientos computacionales. En ese caso, los modelos individuales deben corregirse con los términos energéticos presentados en este estudio.

  Además, esta técnica puede aplicarse a sistemas que interactúan con un entorno desconocido. En este caso, no se conoce el vector de estado de la interacción externa, sino sólo la perturbación producida en el sistema principal. Ambas situaciones pueden ser aprendidas por las redes neuronales port-metriplécticas con una precisión similar a la de las técnicas anteriores.

- Capítulo 7 integra los métodos propuestos en el motor físico de una aplicación de realidad aumentada en sólidos deformables. Este trabajo muestra el rendimiento de las redes neuronales informadas por la termodinámica bajo requerimientos de tiempo real. La demo probada en una malla estructurada y no estructurada muestra un gran rendimiento con framerates estables y una interacción fluida con el usuario. Así, los integradores con información física desarrollados en esta tesis podrían ser potencialmente utilizados en aplicaciones de gemelos digitales e industriales en las que la consistencia física de los resultados es fundamental.

9.2 Trabajo futuro

Esta sección propone varias direcciones de investigación para superar las limitaciones comentadas en la sección de conclusiones de cada capítulo individual. Estas ideas surgieron durante el desarrollo de la tesis, teniendo en cuenta que el aprendizaje automático es un campo de investigación muy activo y productivo.

En términos de interpretabilidad, se podría decir que nuestros modelos son una *caja gris*, ya que siguen utilizando una red neuronal entrenada como unidad de cálculo central. Todavía se puede mejorar, utilizando los últimos avances en regresión simbólica [26, 255], que buscan modelos cerrados parsimoniosos a partir de los datos que puedan aproximar mejor la función objetivo y que además sean más rápidos de evaluar. Es especialmente interesante el uso de la regresión simbólica en redes neuronales de grafos [41], ya que los bloques de nodos y aristas tienen una clara interpretación intuitiva como interacciones entre partículas. Sin embargo, estas técnicas están limitadas si se utilizan capas de normalización y vectores latentes, que no tienen por qué ser interpretables en absoluto.

Se podría introducir un método Runge-Kutta de orden superior en el esquema de integración para obtener una mayor precisión en la solución. Sin embargo, requiere varias pasadas hacia adelante a través de la red neuronal para cada paso de tiempo, aumentando la complejidad del esquema de integración y el proceso de formación. Además, los esquemas de integración basados en GENERIC han mostrado muy buen rendimiento incluso para aproximaciones de primer orden [222]. Un enfoque similar podría obtenerse con el uso de RNNs o *Transformers* [257, 269]. Esta última arquitectura utiliza la atención en lugar de la recurrencia para encontrar patrones temporales en grandes contextos, y recientemente ha aumentado su popularidad debido a su exitoso uso en modelos de lenguaje (LLMs). Este método es conceptualmente similar a un esquema de integración de alto orden con pesos variables basado en el mecanismo de autoatención [29, 232].

Una limitación del presente trabajo es el uso de datos sintéticos en lugar de experimentales. Es posible abrir un campo de investigación para probar los límites de la metodología presentada aplicada a datos reales capturados, y estudiar la influencia del ruido en las mediciones de aplicaciones del mundo real, como gemelos digitales de procesos industriales o entornos de realidad aumentada/virtual en tiempo real. Se puede realizar un análisis más exhaustivo para evaluar la influencia de los datos ruidosos en la evolución del integrador, con el fin de añadir robustez al método e incluso predecir tiempos de simulación más amplios utilizando el aprendizaje incremental [151, 36].

Como se muestra en el Capítulo 4, el algoritmo de *message passing* tiene varias desventajas cuando se manejan grandes dominios que requieren mallas finas. El trabajo futuro puede superar estas limitaciones mediante la combinación de representaciones en grafos con técnicas de reducción del orden del modelo, como los autoencoders

basados en grafos [132] o las arquitecturas de tipo U-net [65, 275]. La idea es sustituir el message passing por varios pasos de cambio de escala, lo que permite que las condiciones de contorno lleguen a todos los nodos del dominio de la simulación a la vez y que se reduzca el número de parámetros de la red neuronal. Esto también puede mitigarse implementando un vector de atención global para que la información dinámica llegue a todos los nodos del dominio de forma instantánea.

Otro tema interesante para ampliar el trabajo es mejorar la generalización y disminuir la cantidad de datos de entrenamiento mediante arquitecturas *equivariantes* [125, 233], que evitan el aumento de datos explotando la invariancia a ciertos grupos como rotaciones $SO(3)$ o transformaciones euclidianas generales $E(3)$.

En el caso de los autoencoders que preservan la estructura en el Capítulo 5, el espacio latente del bloque de reducción también puede ser regularizado utilizando inferencia variacional a través de autoencoders variacionales [129]. Este enfoque Bayesiano es conveniente en casos donde las variables latentes son muestreadas o interpoladas, y conducen a transiciones más suaves entre ellas. Una línea futura de este trabajo es explorar VAEs aplicados a sistemas físicos y estudiar su influencia en la topología del espacio latente y la extrapolabilidad. El uso de arquitecturas basadas en grafos y representaciones implícitas [31] también puede superar la limitación del formato fijo de malla de entrada y mejorar la generalización a mallas variables. Para mejorar el suavizado de la dinámica reconstruida, podría añadirse a las capas de entrada una capa de codificación basada en la frecuencia que transforme las características dinámicas en función de su contenido en frecuencia y mejore el procedimiento de aprendizaje [246].

El uso de redes neuronales basadas en grafos en problemas de elasticidad tiene numerosas similitudes con la teoría peridinámica [239, 240]. Esta formulación no local de la mecánica de sólidos tiene fuertes fundamentos matemáticos y varias ventajas sobre efectos discontinuos, como la fractura, y fuerzas de largo alcance. Podría ser interesante explorar la adaptación de los conceptos peridinámicos en el contexto del aprendizaje profundo geométrico para aprender efectos más complejos como la plasticidad o la fractura.

Las técnicas presentadas en esta tesis ya han sido utilizadas en varias aplicaciones del mundo real, como el estudio de los efectos de oscilación de líquidos en tiempo real [188] con resultados prometedores. Existe una posible línea de investigación futura hacia aplicaciones de realidad aumentada de forma similar al Capítulo 7, en cuanto a optimización de la técnica propuesta y extensión a geometrías más complejas, casos de carga y propiedades constitutivas de los materiales.

Bibliography

[1] Afkham, Babak Maboudi et al. “Structure preserving model reduction of parametric Hamiltonian systems”. In: *SIAM Journal on Scientific Computing* 39.6 (2017), A2616–A2644 (↑ 36).

[2] Ajay, Anurag et al. “Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3066–3073 (↑ 24).

[3] Allam, Zaheer et al. “The metaverse as a virtual form of smart cities: opportunities and challenges for environmental, economic, and social sustainability in urban futures”. In: *Smart Cities* 5.3 (2022), pp. 771–801 (↑ 105).

[4] Allen, Kelsey R et al. “Physical Design using Differentiable Learned Simulators”. In: *arXiv preprint arXiv:2202.00728* (2022) (↑ 26).

[5] Allen-Blanchette, Christine et al. “Lagnetvip: A lagrangian neural network for video prediction”. In: *arXiv preprint arXiv:2010.12932* (2020) (↑ 35).

[6] Alon, Uri et al. “On the bottleneck of graph neural networks and its practical implications”. In: *arXiv preprint arXiv:2006.05205* (2020) (↑ 115).

[7] Arnol’d, Vladimir Igorevich. *Mathematical methods of classical mechanics*. Vol. 60. Springer Science & Business Media, 2013 (↑ 13).

[8] Arnol’d, Vladimir I et al. “Symplectic geometry”. In: *Dynamical systems IV* (2001), pp. 1–138 (↑ 30).

[9] Ayensa-Jiménez, Jacobo et al. “A new reliability-based data-driven approach for noisy experimental data with physical constraints”. In: *Computer Methods in Applied Mechanics and Engineering* 328 (2018), pp. 752–774 (↑ 39).

[10] Badías, Alberto et al. “An augmented reality platform for interactive aerodynamic design and analysis”. In: *International Journal for Numerical Methods in Engineering* 120.1 (2019), pp. 125–138 (↑ 76).

[11] Badlyan, A Moses et al. "Open physical systems: From GENERIC to port-Hamiltonian systems". In: *arXiv preprint arXiv:1804.04064* (2018) (↑ 97).

[12] Bai, Long-Hu et al. "Accurate storm surge forecasting using the encoder–decoder long short term memory recurrent neural network". In: *Physics of Fluids* 34.1 (2022), p. 016601 (↑ 24).

[13] Baird, Leemon. "Residual algorithms: Reinforcement learning with function approximation". In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 30–37 (↑ 5).

[14] Battaglia, Peter W et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018) (↑ 17, 24, 26, 61).

[15] Beattie, Christopher A et al. "Robust port-Hamiltonian representations of passive systems". In: *Automatica* 100 (2019), pp. 182–186 (↑ 96, 97).

[16] Belbute-Peres, Filipe De Avila et al. "Combining differentiable PDE solvers and graph neural networks for fluid flow prediction". In: *international conference on machine learning*. PMLR. 2020, pp. 2402–2411 (↑ 26).

[17] Belbute-Peres, Filipe De Avila et al. "End-to-end differentiable physics for learning and control". In: *Advances in neural information processing systems* 31 (2018) (↑ 17).

[18] Bertalan, Tom et al. "On learning Hamiltonian systems from data". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.12 (2019), p. 121107 (↑ 35).

[19] Betsch, Peter et al. "Variational formulations for large strain thermo-elastodynamics based on the generic formalism". In: *Proceedings of the 6th European Conference on Computational Mechanics, Glasgow, UK*. 2018, pp. 11–15 (↑ 97).

[20] Bharadwaja, BVSS et al. "Physics-Informed Machine Learning and Uncertainty Quantification for Mechanics of Heterogeneous Materials". In: *Integrating Materials and Manufacturing Innovation* (2022), pp. 1–21 (↑ 15).

[21] Bhat, Harish S et al. "Machine learning a molecular Hamiltonian for predicting electron dynamics". In: *International Journal of Dynamics and Control* 8.4 (2020), pp. 1089–1101 (↑ 35).

[22] Bhattoo, Ravinder et al. "Lagrangian neural network with differentiable symmetries and relational inductive bias". In: *arXiv preprint arXiv:2110.03266* (2021) (↑ 35).

[23] Bronstein, Michael M et al. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42 (↑ 17).

[24] Bronstein, Michael M et al. "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478* (2021) (↑ 13, 18).

[25] Brunton, Steven L et al. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022 (↑ 6).

[26] Brunton, Steven L et al. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937 (↑ 39, 121, 125).

[27] Cai, Shengze et al. "DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks". In: *Journal of Computational Physics* 436 (2021), p. 110296 (↑ 16).

[28] Cai, Shengze et al. "Physics-informed neural networks (PINNs) for fluid mechanics: A review". In: *Acta Mechanica Sinica* 37.12 (2021), pp. 1727–1738 (↑ 15).

[29] Cao, Shuhao. "Choose a transformer: Fourier or galerkin". In: *Advances in neural information processing systems* 34 (2021), pp. 24924–24940 (↑ 121, 125).

[30] Chen, Peter Yichen et al. "CROM: Continuous reduced-order modeling of PDEs using implicit neural representations". In: *arXiv preprint arXiv:2206.02607* (2022) (↑ 106).

[31] Chen, Peter Yichen et al. "Model reduction for the material point method via an implicit neural representation of the deformation map". In: *Journal of Computational Physics* 478 (2023), p. 111908 (↑ 122, 126).

[32] Chen, Ricky TQ et al. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018) (↑ 17).

[33] Chen, Samuel Yen-Chi et al. "Quantum convolutional neural networks for high energy physics data analysis". In: *Physical Review Research* 4.1 (2022), p. 013231 (↑ 23).

[34] Chen, Zhijie et al. "Learning Neural Hamiltonian Dynamics: A Methodological Overview". In: *arXiv preprint arXiv:2203.00128* (2022) (↑ 35).

[35] Cho, Kyunghyun et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014) (↑ 24).

[36] Choi, Euntae et al. "Autoencoder-based incremental class learning without retraining on old data". In: *arXiv preprint arXiv:1907.07872* (2019) (↑ 121, 125).

[37] Clifton, Jesse et al. "Q-learning: Theory and applications". In: *Annual Review of Statistics and Its Application* 7 (2020), pp. 279–301 (↑ 5).

[38] Cohen, Jacob et al. *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013 (↑ 5).

[39] Corso, Gabriele et al. "Diffdock: Diffusion steps, twists, and turns for molecular docking". In: *arXiv preprint arXiv:2210.01776* (2022) (↑ 26).

[40] Cortes, Corinna et al. “Support-vector networks”. In: *Machine learning* 20 (1995), pp. 273–297 (↑ 5).

[41] Cranmer, Miles et al. “Discovering symbolic models from deep learning with inductive biases”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17429–17442 (↑ 26, 121, 125).

[42] Cueto, Elias et al. “Thermodynamics of learning physical phenomena”. In: *Archives of Computational Methods in Engineering* (2023), pp. 1–14 (↑ 97).

[43] Cuomo, Salvatore et al. “Scientific machine learning through physics–informed neural networks: Where we are and what’s next”. In: *Journal of Scientific Computing* 92.3 (2022), p. 88 (↑ 6).

[44] Cybenko, George. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314 (↑ 15).

[45] Dandekar, Raj et al. “Bayesian neural ordinary differential equations”. In: *arXiv preprint arXiv:2012.07244* (2020) (↑ 17).

[46] D’Andrea, Antonio et al. “A PhysX-based framework to develop rehabilitation using haptic and virtual reality”. In: *2013 IEEE International Symposium on Industrial Electronics*. IEEE. 2013, pp. 1–6 (↑ 106).

[47] Du, Juan et al. “POD reduced-order unstructured mesh modeling applied to 2D and 3D fluid flow”. In: *Computers & Mathematics with Applications* 65.3 (2013), pp. 362–379 (↑ 76).

[48] E, Weinan. “A proposal on machine learning via dynamical systems”. In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11 (↑ 40).

[49] E, Weinan. “Machine learning and computational mathematics”. In: *arXiv preprint arXiv:2009.14596* (2020) (↑ 96).

[50] Eidnes, Sølve et al. “Port-Hamiltonian Neural Networks with State Dependent Ports”. In: *arXiv preprint arXiv:2206.02660* (2022) (↑ 97, 100).

[51] Espanol, Pep. “Statistical mechanics of coarse-graining”. In: *Novel Methods in Soft Matter Simulations*. Springer, 2004, pp. 69–115 (↑ 34).

[52] Espanol, Pep et al. “Smoothed dissipative particle dynamics”. In: *Physical Review E* 67.2 (2003), p. 026705 (↑ 34).

[53] Eymard, Robert et al. “Finite volume methods”. In: *Handbook of numerical analysis* 7 (2000), pp. 713–1018 (↑ 4).

[54] Farina, Marco et al. “Searching for new physics with deep autoencoders”. In: *Physical Review D* 101.7 (2020), p. 075021 (↑ 16, 76).

[55] Farrell, PE et al. "Conservative interpolation between volume meshes by local Galerkin projection". In: *Computer Methods in Applied Mechanics and Engineering* 200.1-4 (2011), pp. 89–100 (↑ 76).

[56] Fazeli, Nima et al. "Long-horizon prediction and uncertainty propagation with residual point contact learners". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 7898–7904 (↑ 24).

[57] Fefferman, Charles et al. "Testing the manifold hypothesis". In: *Journal of the American Mathematical Society* 29.4 (2016), pp. 983–1049 (↑ 75).

[58] Fey, Matthias et al. "Fast graph representation learning with PyTorch Geometric". In: *arXiv preprint arXiv:1903.02428* (2019) (↑ 111).

[59] Finke, Thorben et al. "Autoencoders for unsupervised anomaly detection in high energy physics". In: *Journal of High Energy Physics* 2021.6 (2021), pp. 1–32 (↑ 16).

[60] Fix, Evelyn et al. "Discriminatory analysis. Nonparametric discrimination: Consistency properties". In: *International Statistical Review/Revue Internationale de Statistique* 57.3 (1989), pp. 238–247 (↑ 5).

[61] Fresca, Stefania et al. "Deep learning-based reduced order models for the real-time simulation of the nonlinear dynamics of microstructures". In: *International Journal for Numerical Methods in Engineering* 123.20 (2022), pp. 4749–4777 (↑ 106).

[62] Fulton, Lawson et al. "Latent-space dynamics for reduced deformable simulation". In: *Computer graphics forum*. Vol. 38. 2. Wiley Online Library. 2019, pp. 379–391 (↑ 106).

[63] Galimberti, Clara Lucía et al. "A unified framework for Hamiltonian deep neural networks". In: *Learning for Dynamics and Control*. PMLR. 2021, pp. 275–286 (↑ 35).

[64] Galley, Chad R et al. "The principle of stationary nonconservative action for classical mechanics and field theories". In: *arXiv preprint arXiv:1412.3082* (2014) (↑ 26).

[65] Gao, Hongyang et al. "Graph u-nets". In: *international conference on machine learning*. PMLR. 2019, pp. 2083–2092 (↑ 121, 126).

[66] Gao, Yuanqi et al. "Harnessing deep reinforcement learning to construct time-dependent optimal fields for quantum control dynamics". In: *Physical Chemistry Chemical Physics* 24.39 (2022), pp. 24012–24020 (↑ 35).

[67] Ghavamian, F et al. "Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network". In: *Computer Methods in Applied Mechanics and Engineering* 357 (2019), p. 112594 (↑ 24).

[68] Ghnatios, Chady et al. "Data-driven generic modeling of poroviscoelastic materials". In: *Entropy* 21.12 (2019), p. 1165 (↑ 40).

[69] Gilmer, Justin et al. "Neural message passing for quantum chemistry". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1263–1272 (↑ 26, 62).

[70] Gong, Yuezheng et al. "Structure-preserving Galerkin POD reduced-order modeling of Hamiltonian systems". In: *Computer Methods in Applied Mechanics and Engineering* 315 (2017), pp. 780–798 (↑ 36).

[71] Gonsalves, Tad. "The summers and winters of artificial intelligence". In: *Advanced methodologies and technologies in artificial intelligence, computer simulation, and human-computer interaction*. IGI Global, 2019, pp. 168–179 (↑ 4).

[72] González, David et al. "Consistent data-driven computational mechanics". In: *AIP Conference Proceedings*. Vol. 1960. 1. AIP Publishing LLC. 2018, p. 090005 (↑ 40, 92).

[73] González, David et al. "Learning corrections for hyperelastic models from data". In: *Frontiers in Materials* 6 (2019), p. 14 (↑ 40).

[74] González, David et al. "Thermodynamically consistent data-driven computational mechanics". In: *Continuum Mechanics and Thermodynamics* 31.1 (2019), pp. 239–253 (↑ 41, 42, 47, 56, 57, 84).

[75] Gonzalez, Francisco J et al. "Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems". In: *arXiv preprint arXiv:1808.01346* (2018) (↑ 23).

[76] Goodfellow, Ian et al. *Deep learning*. MIT press, 2016 (↑ 13, 16, 76).

[77] Goodfellow, Ian et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144 (↑ 23).

[78] Grmela, Miroslav et al. "Dynamics and thermodynamics of complex fluids. I. Development of a general formalism". In: *Physical Review E* 56.6 (1997), p. 6620 (↑ 31, 40).

[79] Grmela, Miroslav et al. "Gradient and GENERIC time evolution towards reduced dynamics". In: *Philosophical Transactions of the Royal Society A* 378.2170 (2020), p. 20190472 (↑ 80).

[80] Grmela, Miroslav et al. "Nonlinear and Hamiltonian extended irreversible thermodynamics". In: *The Journal of chemical physics* 108.19 (1998), pp. 7937–7945 (↑ 34).

[81] Gruber, Anthony et al. "Energetically consistent model reduction for metriplectic systems". In: *Computer Methods in Applied Mechanics and Engineering* 404 (2023), p. 115709 (↑ 36).

[82] Gruver, Nate et al. "Deconstructing The Inductive Biases Of Hamiltonian Neural Networks". In: *arXiv preprint arXiv:2202.04836* (2022) (↑ 97).

[83] Guillemin, Victor et al. *Symplectic techniques in physics*. Cambridge university press, 1990 (↑ 30).

[84] Guo, Mengwu et al. "Energy-Based Error Bound of Physics-Informed Neural Network Solutions in Elasticity". In: *Journal of Engineering Mechanics* 148.8 (2022), p. 04022038 (↑ 15).

[85] Haghighat, Ehsan et al. "A nonlocal physics-informed deep learning framework using the peridynamic differential operator". In: *Computer Methods in Applied Mechanics and Engineering* 385 (2021), p. 114012 (↑ 15).

[86] Haghighat, Ehsan et al. "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 379 (2021), p. 113741 (↑ 15).

[87] Haghighat, Ehsan et al. "Constitutive model characterization and discovery using physics-informed deep learning". In: *Engineering Applications of Artificial Intelligence* 120 (2023), p. 105828 (↑ 15).

[88] Hamilton, Will et al. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017) (↑ 26).

[89] Han, Jiaqi et al. "Learning physical dynamics with subequivariant graph neural networks". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 26256–26268 (↑ 97).

[90] Hartigan, John A et al. "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), pp. 100–108 (↑ 5).

[91] He, Kaiming et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (↑ 62).

[92] He, Kaiming et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. ICCV. 2015, pp. 1026–1034 (↑ 48, 54, 84, 88).

[93] He, Xiaolong et al. "Deep autoencoders for physics-constrained data-driven nonlinear materials modeling". In: *Computer Methods in Applied Mechanics and Engineering* 385 (2021), p. 114034 (↑ 16).

[94] Heiden, Eric et al. "NeuralSim: Augmenting differentiable simulators with neural networks". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 9474–9481 (↑ 16).

[95] Hernandez, Quercus et al. "Deep learning of thermodynamics-aware reduced-order models from data". In: *Computer Methods in Applied Mechanics and Engineering* 379 (2021), p. 113763 (↑ 39, 75).

[96] Hernández, Quercus et al. "Port-metriplectic neural networks: thermodynamics-informed machine learning of complex physical systems". In: *Computational Mechanics* (2023), pp. 1–9 (↑ 95).

[97] Hernández, Quercus et al. "Structure-preserving neural networks". In: *Journal of Computational Physics* 426 (2021), p. 109950 (↑ 92, 102).

[98] Hernández, Quercus et al. "Thermodynamics-informed graph neural networks". In: *arXiv preprint arXiv:2203.01874* (2022) (↑ 59).

[99] Hernández, Quercus et al. "Thermodynamics-informed neural networks for physically realistic mixed reality". In: *Computer Methods in Applied Mechanics and Engineering* 407 (2023), p. 115912 (↑ 105).

[100] Hesthaven, Jan S et al. "Structure-preserving model order reduction of Hamiltonian systems". In: *arXiv preprint arXiv:2109.12367* (2021) (↑ 36).

[101] Hey, Anthony JG et al. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA, 2009 (↑ 4).

[102] Ho, Tin Kam. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282 (↑ 5).

[103] Hochreiter, Sepp et al. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780 (↑ 24).

[104] Hohenberg, Pierre C et al. "Theory of dynamic critical phenomena". In: *Reviews of Modern Physics* 49.3 (1977), p. 435 (↑ 96).

[105] Hornik, Kurt. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257 (↑ 64).

[106] Hornik, Kurt et al. "Multilayer feedforward networks are universal approximators." In: *Neural networks* 2.5 (1989), pp. 359–366 (↑ 15).

[107] Hu, Weihua et al. "Forcenet: A graph neural network for large-scale quantum calculations". In: *arXiv preprint arXiv:2103.01436* (2021) (↑ 26).

[108] Huang, Shenglin et al. "Variational Onsager Neural Networks (VONNs): A thermodynamics-based variational learning strategy for non-equilibrium PDEs". In: *Journal of the Mechanics and Physics of Solids* 163 (2022), p. 104856 (↑ 36).

[109] Hughes, Gordon. "On the mean accuracy of statistical pattern recognizers". In: *IEEE transactions on information theory* 14.1 (1968), pp. 55–63 (↑ 17).

[110] Hughes, Thomas JR. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012 (↑ 4).

[111] Hummel, Mathias et al. "Leveraging nvidia omniverse for in situ visualization". In: *International Conference on High Performance Computing*. Springer. 2019, pp. 634–642 (↑ 106).

[112] Ibáñez, Rubén et al. "Hybrid constitutive modeling: data-driven learning of corrections to plasticity models". In: *International Journal of Material Forming* 12.4 (2019), pp. 717–725 (↑ 39).

[113] Ibañez, Rubén et al. "Data-driven non-linear elasticity: constitutive manifold construction and problem discretization". In: *Computational Mechanics* 60.5 (2017), pp. 813–826 (↑ 39).

[114] Ibañez, Rubén et al. "A manifold learning approach to data-driven computational elasticity and inelasticity". In: *Archives of Computational Methods in Engineering* 25.1 (2018), pp. 47–57 (↑ 39).

[115] Jiang, Zhongyi et al. "Fourier-MIONet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration". In: *arXiv preprint arXiv:2303.04778* (2023) (↑ 16).

[116] Jin, Pengzhan et al. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132 (2020), pp. 166–179 (↑ 35).

[117] Jin, Xiaowei et al. "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations". In: *Journal of Computational Physics* 426 (2021), p. 109951 (↑ 15).

[118] Jin, Xiaowei et al. "Time-resolved reconstruction of flow field around a circular cylinder by recurrent neural networks based on non-time-resolved particle image velocimetry measurements". In: *Experiments in Fluids* 61 (2020), pp. 1–23 (↑ 24).

[119] Ju, Xiangyang et al. "Graph neural networks for particle reconstruction in high energy physics detectors". In: *arXiv preprint arXiv:2003.11603* (2020) (↑ 26).

[120] Karniadakis, George Em et al. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440 (↑ 6, 15).

[121] Kashinath, Karthik et al. "Physics-informed machine learning: case studies for weather and climate modelling". In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200093 (↑ 6).

[122] Kaufman, Allan N. "Dissipative Hamiltonian systems: A unifying principle". In: *Physics Letters A* 100.8 (1984), pp. 419–422 (↑ 31).

[123] Kaufman, Allan N. "Lorentz-covariant dissipative Lagrangian systems". In: *Physics Letters A* 109.3 (1985), pp. 87–89 (↑ 34).

[124] Kelly, Jacob et al. "Learning differential equations that are easy to solve". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4370–4380 (↑ 17).

[125] Keriven, Nicolas et al. "Universal invariant and equivariant graph neural networks". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 7092–7101 (↑ 122, 126).

[126] Kidger, Patrick et al. "Universal approximation with deep narrow networks". In: *Conference on learning theory*. PMLR. 2020, pp. 2306–2327 (↑ 15).

[127] Kim, Suyong et al. "Stiff neural ordinary differential equations". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.9 (2021), p. 093122 (↑ 17).

[128] Kingma, Diederik P et al. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014) (↑ 48, 54, 65, 84, 88).

[129] Kingma, Diederik P et al. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013) (↑ 122, 126).

[130] Kipf, Thomas N et al. "Neural relational inference for interacting systems". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2688–2697 (↑ 26).

[131] Kipf, Thomas N et al. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016) (↑ 26).

[132] Kipf, Thomas N et al. "Variational graph auto-encoders". In: *arXiv preprint arXiv:1611.07308* (2016) (↑ 121, 126).

[133] Kirchdoerfer, Trenton et al. "Data-driven computational mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 304 (2016), pp. 81–101 (↑ 6, 39).

[134] Kloss, Alina et al. "Combining learned and analytical models for predicting action effects". In: *arXiv preprint arXiv:1710.04102* 11 (2017) (↑ 23).

[135] Kochkov, Dmitrii et al. "Learning ground states of quantum Hamiltonians with graph networks". In: *arXiv preprint arXiv:2110.06390* (2021) (↑ 35).

[136] Kraus, Sascha et al. "Facebook and the creation of the metaverse: radical business model innovation or incremental transformation?" In: *International Journal of Entrepreneurial Behavior & Research* (2022) (↑ 106).

[137] Kubo, Rep. "The fluctuation-dissipation theorem". In: *Reports on progress in physics* 29.1 (1966), p. 255 (↑ 34).

[138] Laso, Manuel et al. “Calculation of viscoelastic flow using molecular models: the CONNFFESSIT approach”. In: *Journal of Non-Newtonian Fluid Mechanics* 47 (1993), pp. 1–20 (↑ 52, 66).

[139] Le Bris, Claude et al. “Multiscale modelling of complex fluids: a mathematical initiation”. In: *Multiscale modeling and simulation in science*. Springer, 2009, pp. 49–137 (↑ 52, 54, 66, 83).

[140] LeCun, Yann et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (↑ 23).

[141] LeCun, Yann et al. “Object recognition with gradient-based learning”. In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345 (↑ 17).

[142] Lee, Honglak et al. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”. In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 609–616 (↑ 14).

[143] Lee, Kookjin et al. “Machine learning structure preserving brackets for forecasting irreversible processes”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5696–5707 (↑ 36, 64).

[144] Lee, Kookjin et al. “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders”. In: *Journal of Computational Physics* 404 (2020), p. 108973 (↑ 16, 76, 77).

[145] Lee, Seung-Chul et al. “An enhanced Lagrangian neural network for the ELD problems with piecewise quadratic cost functions and nonlinear constraints”. In: *Electric Power Systems Research* 60.3 (2002), pp. 167–177 (↑ 35).

[146] Leoni, Patricio Clark di et al. “Deep operator neural networks (DeepONets) for prediction of instability waves in high-speed boundary layers”. In: *APS Division of Fluid Dynamics Meeting Abstracts*. 2020, R01–004 (↑ 16).

[147] Leshno, Moshe et al. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural Networks* 6.6 (1993), pp. 861–867 (↑ 15).

[148] Li, Xiao et al. “Exploring NVIDIA Omniverse for Future Space Resources Missions”. In: (2022) (↑ 106).

[149] Li, Xuan et al. “Physics-informed deep learning model in wind turbine response prediction”. In: *Renewable Energy* 185 (2022), pp. 932–944 (↑ 24).

[150] Li, Yunzhu et al. “Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids”. In: *arXiv preprint arXiv:1810.01566* (2018) (↑ 26).

[151] Li, Zhizhong et al. “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947 (↑ 121, 125).

[152] Li, Zongyi et al. “Fourier neural operator for parametric partial differential equations”. In: *arXiv preprint arXiv:2010.08895* (2020) (↑ 16).

[153] Liu, Qi et al. “Constrained graph variational autoencoders for molecule design”. In: *Advances in neural information processing systems*. 2018, pp. 7795–7804 (↑ 16, 76).

[154] Liu, Xin et al. “A review of artificial neural networks in the constitutive modeling of composite materials”. In: *Composites Part B: Engineering* 224 (2021), p. 109152 (↑ 15).

[155] Liu, Yuying et al. “Multiresolution convolutional autoencoders”. In: *Journal of Computational Physics* 474 (2023), p. 111801 (↑ 23).

[156] Liu, Ziming et al. “Physics-augmented learning: A new paradigm beyond physics-informed learning”. In: *arXiv preprint arXiv:2109.13901* (2021) (↑ 6).

[157] Long, Jonathan et al. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440 (↑ 23).

[158] Lu, Lu et al. “Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators”. In: *arXiv preprint arXiv:1910.03193* (2019) (↑ 15).

[159] Lu, Lu et al. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature machine intelligence* 3.3 (2021), pp. 218–229 (↑ 15).

[160] Lu, Lu et al. “Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport”. In: *Physical Review Research* 4.2 (2022), p. 023210 (↑ 16).

[161] Ludwig, Wolfgang et al. *Symmetries in physics: group theory applied to physical problems*. Vol. 64. Springer Science & Business Media, 2012 (↑ 17, 26).

[162] Lusch, Bethany et al. “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature communications* 9.1 (2018), p. 4950 (↑ 16).

[163] Lutter, Michael et al. “Deep lagrangian networks: Using physics as model prior for deep learning”. In: *arXiv preprint arXiv:1907.04490* (2019) (↑ 35).

[164] Ma, Hao et al. “A combined data-driven and physics-driven method for steady heat conduction prediction using deep convolutional neural networks”. In: *arXiv preprint arXiv:2005.08119* (2020) (↑ 23).

[165] Maboudi Afkham, Babak et al. “Structure-preserving model-reduction of dissipative Hamiltonian systems”. In: *Journal of Scientific Computing* 81.1 (2019), pp. 3–21 (↑ 36).

[166] Machalek, Derek et al. "Dynamic energy system modeling using hybrid physics-based and machine learning encoder–decoder models". In: *Energy and AI* 9 (2022), p. 100172 (↑ 24).

[167] Maciel, Anderson et al. "Using the PhysX engine for physics-based virtual surgery with force feedback". In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 5.3 (2009), pp. 341–353 (↑ 106).

[168] Magill, Martin et al. "Neural networks trained to solve differential equations learn general representations". In: *Advances in neural information processing systems* 31 (2018) (↑ 15).

[169] Mao, Zhiping et al. "DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators". In: *Journal of computational physics* 447 (2021), p. 110698 (↑ 16).

[170] Marco, Julio et al. "DeepToF: off-the-shelf real-time correction of multipath interference in time-of-flight imaging". In: *ACM Transactions on Graphics (ToG)* 36.6 (2017), pp. 1–12 (↑ 76).

[171] Martin-Brualla, Ricardo et al. "Lookingood: Enhancing performance capture with real-time neural re-rendering". In: *arXiv preprint arXiv:1811.05029* (2018) (↑ 115).

[172] Masci, Jonathan et al. "Stacked convolutional auto-encoders for hierarchical feature extraction". In: *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*. Springer. 2011, pp. 52–59 (↑ 23).

[173] Materassi, Massimo et al. "Metriplectic torque for rotation control of a rigid body". In: *arXiv preprint arXiv:1807.01168* (2018) (↑ 34).

[174] Messner, Mark C. "Convolutional neural network surrogate models for the mechanical properties of periodic structures". In: *Journal of Mechanical Design* 142.2 (2020), p. 024503 (↑ 23).

[175] Mielke, Alexander. "Formulation of thermoelastic dissipative material behavior using GENERIC". In: *Continuum Mechanics and Thermodynamics* 23.3 (2011), pp. 233–256 (↑ 34).

[176] Miller, Scott T et al. "Mastering high-dimensional dynamics with Hamiltonian neural networks". In: *arXiv preprint arXiv:2008.04214* (2020) (↑ 35).

[177] Misra, Diganta. "Mish: A self regularized non-monotonic neural activation function". In: *arXiv preprint arXiv:1908.08681* 4 (2019) (↑ 64).

[178] Mitchell, Tom M. *The need for biases in learning generalizations*. Citeseer, 1980 (↑ 18).

[179] Mohajerin, Nima et al. "Multistep prediction of dynamic systems with recurrent neural networks". In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3370–3383 (↑ 24).

[180] Montefusco, Alberto et al. "Essential equivalence of the general equation for the nonequilibrium reversible-irreversible coupling (GENERIC) and steepest-entropy-ascent models of dissipation for nonequilibrium thermodynamics". In: *Physical Review E* 91.4 (2015), p. 042138 (↑ 33).

[181] Monti, Federico et al. "Dual-primal graph convolutional networks". In: *arXiv preprint arXiv:1806.00770* (2018) (↑ 60).

[182] Mori, Hazime. "Transport, collective motion, and Brownian motion". In: *Progress of theoretical physics* 33.3 (1965), pp. 423–455 (↑ 34).

[183] Morrison, Philip J. "A paradigm for joined Hamiltonian and dissipative systems". In: *Physica D: Nonlinear Phenomena* 18.1-3 (1986), pp. 410–419 (↑ 31, 34).

[184] Morrison, Philip J. "Bracket formulation for irreversible classical fields". In: *Physics Letters A* 100.8 (1984), pp. 423–427 (↑ 31).

[185] Morrison, Philip J. "Some observations regarding brackets and dissipation". In: *Center for Pure and Applied Mathematics Report PAM-228, University of California, Berkeley* (1984) (↑ 32, 34).

[186] Moya, Beatriz et al. "Learning slosh dynamics by means of data". In: *Computational Mechanics* 64.2 (2019), pp. 511–523 (↑ 76).

[187] Moya, Beatriz et al. "Physically sound, self-learning digital twins for sloshing fluids". In: *PLOS ONE* 15.6 (2020), e0234569 (↑ 76).

[188] Moya, Beatriz et al. "Physics perception in sloshing scenes with guaranteed thermodynamic consistency". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.2 (2022), pp. 2136–2150 (↑ 122, 126).

[189] Murdoch, W James et al. "Interpretable machine learning: definitions, methods, and applications". In: *arXiv preprint arXiv:1901.04592* (2019) (↑ 40).

[190] Mystakidis, Stylianos. "Metaverse". In: *Encyclopedia* 2.1 (2022), pp. 486–497 (↑ 106).

[191] Nabian, Mohammad Amin et al. "Physics-driven regularization of deep neural networks for enhanced engineering design and analysis". In: *Journal of Computing and Information Science in Engineering* 20.1 (2020), p. 011006 (↑ 15).

[192] Ng, Andrew et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19 (↑ 78).

[193] Niaki, Sina Amini et al. "Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture". In: *Computer Methods in Applied Mechanics and Engineering* 384 (2021), p. 113959 (↑ 15).

[194] Niroomandi, Siamak et al. "Real-time deformable models of non-linear tissues by model reduction techniques". In: *Computer methods and programs in biomedicine* 91.3 (2008), pp. 223–231 (↑ 76).

[195] Obayashi, Wataru et al. "Feature extraction of fields of fluid dynamics data using sparse convolutional autoencoder". In: *AIP Advances* 11.10 (2021), p. 105211 (↑ 23).

[196] Odot, Alban et al. "DeepPhysics: A physics aware deep learning framework for real-time simulation". In: *International Journal for Numerical Methods in Engineering* 123.10 (2022), pp. 2381–2398 (↑ 106).

[197] Öttinger, Hans Christian. *Beyond equilibrium thermodynamics*. John Wiley & Sons, 2005 (↑ 13).

[198] Öttinger, Hans Christian. "Nonequilibrium thermodynamics for open systems". In: *Physical Review E* 73.3 (2006), p. 036126 (↑ 97, 98).

[199] Öttinger, Hans Christian. "Preservation of thermodynamic structure in model reduction". In: *Physical Review E* 91.3 (2015), p. 032147 (↑ 78).

[200] Öttinger, Hans Christian et al. "Dynamics and thermodynamics of complex fluids. II. Illustrations of a general formalism". In: *Physical Review E* 56.6 (1997), p. 6633 (↑ 31, 34).

[201] Otto, Samuel E et al. "Linearly recurrent autoencoder networks for learning dynamics". In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 558–593 (↑ 16).

[202] Owens, Robert G et al. *Computational rheology*. World Scientific, 2002 (↑ 52).

[203] Parger, Mathias et al. "UNOC: Understanding occlusion for embodied presence in virtual reality". In: *IEEE Transactions on Visualization and Computer Graphics* (2021) (↑ 115).

[204] Paszke, Adam et al. "Automatic differentiation in pytorch". In: (2017) (↑ 44, 62, 79, 82).

[205] Paszke, Adam et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019) (↑ 44).

[206] Peng, Liqian et al. "Symplectic model reduction of Hamiltonian systems". In: *SIAM Journal on Scientific Computing* 38.1 (2016), A1–A27 (↑ 36).

[207] Pfaff, Tobias et al. "Learning mesh-based simulation with graph networks". In: *arXiv preprint arXiv:2010.03409* (2020) (↑ 26, 64, 65).

[208] Pfrommer, Samuel et al. "Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations". In: *Conference on Robot Learning*. PMLR. 2021, pp. 2279–2291 (↑ 16).

[209] Pinkus, Allan. "Approximation theory of the MLP model in neural networks". In: *Acta numerica* 8 (1999), pp. 143–195 (↑ 15).

[210] Prud'Homme, Christophe et al. "Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods". In: *J. Fluids Eng.* 124.1 (2002), pp. 70–80 (↑ 76).

[211] Psihas, Fernanda et al. "Context-enriched identification of particles with a convolutional network for neutrino events". In: *Physical Review D* 100.7 (2019), p. 073005 (↑ 23).

[212] Qiao, Yi-Ling et al. "Scalable differentiable physics for learning and control". In: *arXiv preprint arXiv:2007.02168* (2020) (↑ 16).

[213] Rahaman, Nasim et al. "On the spectral bias of neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5301–5310 (↑ 92).

[214] Raissi, Maziar et al. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707 (↑ 15).

[215] Ramachandran, Prajit et al. "Searching for activation functions". In: *arXiv preprint arXiv:1710.05941* (2017) (↑ 64).

[216] Rao, Chengping et al. "Physics-informed deep learning for computational elastodynamics without labeled data". In: *Journal of Engineering Mechanics* 147.8 (2021), p. 04021043 (↑ 15).

[217] Rashad, Ramy et al. "Twenty years of distributed port-Hamiltonian systems: a literature review". In: *IMA Journal of Mathematical Control and Information* 37.4 (2020), pp. 1400–1422 (↑ 96, 97).

[218] Reyes, Brandon et al. "Learning unknown physics of non-Newtonian fluids". In: *Physical Review Fluids* 6.7 (2021), p. 073301 (↑ 15).

[219] Romero, Cristian et al. "Contact-centric deformation learning". In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–11 (↑ 106).

[220] Romero, Ignacio. "Algorithms for coupled problems that preserve symmetries and the laws of thermodynamics: Part I: Monolithic integrators and their application to finite strain thermoelasticity". In: *Computer Methods in Applied Mechanics and Engineering* 199.25-28 (2010), pp. 1841–1858 (↑ 34).

[221] Romero, Ignacio. "Algorithms for coupled problems that preserve symmetries and the laws of thermodynamics: Part II: Fractional step methods". In: *Computer Methods in Applied Mechanics and Engineering* 199.33-36 (2010), pp. 2235–2248 (↑ 34).

[222] Romero, Ignacio. "Thermodynamically consistent time-stepping algorithms for non-linear thermomechanical systems". In: *International journal for numerical methods in engineering* 79.6 (2009), pp. 706–732 (↑ 47, 100, 121, 125).

[223] Romero, Javier et al. "Embodied hands: Modeling and capturing hands and bodies together". In: *arXiv preprint arXiv:2201.02610* (2022) (↑ 106).

[224] Rospigliosi, Pericles 'asher'. *Metaverse or Simulacra? Roblox, Minecraft, Meta and the turn to virtual reality for education, socialisation and work*. 2022 (↑ 105).

[225] Rowley, Clarence W et al. "Model reduction for compressible flows using POD and Galerkin projection". In: *Physica D: Nonlinear Phenomena* 189.1-2 (2004), pp. 115–129 (↑ 76).

[226] Ruder, Sebastian. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016) (↑ 44, 79, 82).

[227] Rumelhart, David E et al. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (↑ 17).

[228] Russell, Stuart J. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010 (↑ 4).

[229] Salmela, Lauri et al. "Predicting ultrafast nonlinear dynamics in fibre optics with a recurrent neural network". In: *Nature machine intelligence* 3.4 (2021), pp. 344–354 (↑ 24).

[230] Sanchez-Gonzalez, Alvaro et al. "Hamiltonian graph networks with ode integrators". In: *arXiv preprint arXiv:1909.12790* (2019) (↑ 35).

[231] Sanchez-Gonzalez, Alvaro et al. "Learning to simulate complex physics with graph networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8459–8468 (↑ 26, 65).

[232] Sanchis-Agudo, Marcial et al. "Easy attention: A simple self-attention mechanism for Transformers". In: *arXiv preprint arXiv:2308.12874* (2023) (↑ 121, 125).

[233] Satorras, Victor Garcia et al. "E(n) equivariant graph neural networks". In: *International conference on machine learning*. PMLR. 2021, pp. 9323–9332 (↑ 122, 126).

[234] Schuster, Mike et al. "Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681 (↑ 24).

[235] Seifert, Udo et al. "Fluctuation-dissipation theorem in nonequilibrium steady states". In: *Europhysics Letters* 89.1 (2010), p. 10007 (↑ 34).

[236] Shen, Siyuan et al. "High-order differentiable autoencoder for nonlinear model reduction". In: *arXiv preprint arXiv:2102.11026* (2021) (↑ 16).

[237] Shi, Weijing et al. "Point-gnn: Graph neural network for 3d object detection in a point cloud". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 1711–1719 (↑ 60).

[238] Shlomi, Jonathan et al. "Graph neural networks in particle physics". In: *Machine Learning: Science and Technology* 2.2 (2020), p. 021001 (↑ 26).

[239] Silling, Stewart A. "Reformulation of elasticity theory for discontinuities and long-range forces". In: *Journal of the Mechanics and Physics of Solids* 48.1 (2000), pp. 175–209 (↑ 122, 126).

[240] Silling, Stewart A et al. "Peridynamic theory of solid mechanics". In: *Advances in applied mechanics* 44 (2010), pp. 73–168 (↑ 122, 126).

[241] Sohl-Dickstein, Jascha et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265 (↑ 5).

[242] Solera-Rico, Alberto et al. "beta-Variational autoencoders and transformers for reduced-order modelling of fluid flows". In: *arXiv preprint arXiv:2304.03571* (2023) (↑ 23).

[243] Sutskever, Ilya et al. "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. PMLR. 2013, pp. 1139–1147 (↑ 18).

[244] Takeishi, Naoya et al. "Learning Koopman invariant subspaces for dynamic mode decomposition". In: *Advances in neural information processing systems* 30 (2017) (↑ 16).

[245] Tallec, Corentin et al. "Can recurrent neural networks warp time?" In: *arXiv preprint arXiv:1804.11188* (2018) (↑ 24).

[246] Tancik, Matthew et al. "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7537–7547 (↑ 122, 126).

[247] Teichert, Gregory H et al. "Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions". In: *Computer Methods in Applied Mechanics and Engineering* 353 (2019), pp. 201–216 (↑ 62).

[248] Tompson, Jonathan et al. "Accelerating eulerian fluid simulation with convolutional networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3424–3433 (↑ 23).

[249] Tong, Yunjin et al. "Symplectic neural networks in Taylor series form for Hamiltonian systems". In: *Journal of Computational Physics* 437 (2021), p. 110325 (↑ 35).

[250] Topping, Jake et al. "Understanding over-squashing and bottlenecks on graphs via curvature". In: *arXiv preprint arXiv:2111.14522* (2021) (↑ 115).

[251] Toth, Peter et al. "Hamiltonian generative networks". In: *arXiv preprint arXiv: 1909.13789* (2019) (↑ 35).

[252] Trischler, Adam P et al. "Synthesis of recurrent neural networks for dynamical system simulation". In: *Neural Networks* 80 (2016), pp. 67–78 (↑ 24).

[253] Turner, M Jon et al. "Stiffness and deflection analysis of complex structures". In: *journal of the Aeronautical Sciences* 23.9 (1956), pp. 805–823 (↑ 4).

[254] Turski, Lukasz A et al. "Canonical-dissipative formulation of relativistic plasma kinetic theory with self-consistent Maxwell field". In: *Physics Letters A* 120.7 (1987), pp. 331–333 (↑ 34).

[255] Udrescu, Silviu-Marian et al. "AI Feynman: A physics-inspired method for symbolic regression". In: *Science Advances* 6.16 (2020), eaay2631 (↑ 121, 125).

[256] Van Der Schaft, Arjan et al. "Port-Hamiltonian systems theory: An introductory overview". In: *Foundations and Trends® in Systems and Control* 1.2-3 (2014), pp. 173–378 (↑ 96, 97).

[257] Vaswani, Ashish et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017) (↑ 24, 121, 125).

[258] Veeraiah, Vivek et al. "Enhancement of Meta Verse Capabilities by IoT Integration". In: *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE. 2022, pp. 1493–1498 (↑ 105).

[259] Veličković, Petar et al. "Graph attention networks". In: *arXiv preprint arXiv: 1710.10903* (2017) (↑ 26, 60).

[260] Vinuesa, Ricardo et al. "Enhancing computational fluid dynamics with machine learning". In: *Nature Computational Science* 2.6 (2022), pp. 358–366 (↑ 23).

[261] Wang, Fei-Yue et al. "Metasocieties in metaverse: Metaeconomics and metamanagement for metaenterprises and metacities". In: *IEEE Transactions on Computational Social Systems* 9.1 (2022), pp. 2–7 (↑ 105).

[262] Wang, Hanchen et al. "Scientific discovery in the age of artificial intelligence". In: *Nature* 620.7972 (2023), pp. 47–60 (↑ 6).

[263] Wang, Jia-Ji et al. "A deep learning framework for constitutive modeling based on temporal convolutional network". In: *Journal of Computational Physics* 449 (2022), p. 110784 (↑ 23).

[264] Wang, Lei et al. "A method for 3D rock fracturing simulation based Havok". In: *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE. 2015, pp. 898–901 (↑ 106).

[265] Wang, Rui et al. "Approximately equivariant networks for imperfectly symmetric dynamics". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23078–23091 (↑ 97).

[266] Wang, Sifan et al. "Learning the solution operator of parametric partial differential equations with physics-informed deeponets". In: *Science advances* 7.40 (2021), eabi8605 (↑ 16).

[267] Wang, Yi-Jen et al. "Runge-Kutta neural network for identification of dynamical systems in high accuracy". In: *IEEE Transactions on Neural Networks* 9.2 (1998), pp. 294–307 (↑ 121).

[268] Wang, Yue et al. "Dynamic graph cnn for learning on point clouds". In: *Acm Transactions On Graphics (tog)* 38.5 (2019), pp. 1–12 (↑ 60).

[269] Wen, Qingsong et al. "Transformers in time series: A survey". In: *arXiv preprint arXiv:2202.07125* (2022) (↑ 121, 125).

[270] Wold, Svante et al. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52 (↑ 5).

[271] Yan, Wei. "Augmented reality instructions for construction toys enabled by accurate model registration and realistic object/hand occlusions". In: *Virtual Reality* 26.2 (2022), pp. 465–478 (↑ 115).

[272] Yang, Tsung-Yen et al. "Learning physics constrained dynamics using autoencoders". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17157–17172 (↑ 16).

[273] Yasnitsky, Leonid N. "Whether be new "Winter" of artificial intelligence?" In: *Integrated Science in Digital Age: ICIS 2019*. Springer. 2020, pp. 13–17 (↑ 4).

[274] Ye, Huanchun et al. "Poisson bracket for the Vlasov equation on a symplectic leaf". In: *Physics Letters A* 156.1-2 (1991), pp. 96–100 (↑ 34).

[275] Yu, Bing et al. "St-unet: A spatio-temporal u-network for graph-structured time series modeling". In: *arXiv preprint arXiv:1903.05631* (2019) (↑ 121, 126).

[276] Yu, Guoshen et al. "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity". In: *IEEE Transactions on Image Processing* 21.5 (2011), pp. 2481–2499 (↑ 5).

[277] Yu, Haijun et al. "OnsagerNet: Learning stable and interpretable dynamics using a generalized Onsager principle". In: *Physical Review Fluids* 6.11 (2021), p. 114402 (↑ 36).

[278] Zhang, Fan et al. "Mediapipe hands: On-device real-time hand tracking". In: *arXiv preprint arXiv:2006.10214* (2020) (↑ 111).

[279] Zhang, Jiani et al. "Gaan: Gated attention networks for learning on large and spatiotemporal graphs". In: *arXiv preprint arXiv:1803.07294* (2018) (↑ 60).

[280] Zhang, Ruiyang et al. "Physics-informed multi-LSTM networks for metamodeling of nonlinear structures". In: *Computer Methods in Applied Mechanics and Engineering* 369 (2020), p. 113226 (↑ 24).

[281] Zhang, Xuan et al. "Artificial Intelligence for Science in Quantum, Atomistic, and Continuum Systems". In: *arXiv preprint arXiv:2307.08423* (2023) (↑ 6).

[282] Zhang, Zhen et al. "GFINNs: GENERIC formalism informed neural networks for deterministic and stochastic dynamical systems". In: *Philosophical Transactions of the Royal Society A* 380.2229 (2022), p. 20210207 (↑ 36, 64).

[283] Zhao, Xiaoyu et al. "Physics-informed convolutional neural networks for temperature field prediction of heat source layout without labeled data". In: *Engineering Applications of Artificial Intelligence* 117 (2023), p. 105516 (↑ 23).

[284] Zheng, Mianlun et al. "A Deep Emulator for Secondary Motion of 3D Characters". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5932–5940 (↑ 26).

[285] Zhong, Yaofeng Desmond et al. "Benchmarking energy-conserving neural networks for learning dynamics from data". In: *Learning for Dynamics and Control*. PMLR. 2021, pp. 1218–1229 (↑ 96).

[286] Zhong, Yaofeng Desmond et al. "Dissipative symoden: Encoding hamiltonian dynamics with dissipation and control into deep learning". In: *arXiv preprint arXiv:2002.08860* (2020) (↑ 96).

[287] Zhong, Yaofeng Desmond et al. "Unsupervised learning of lagrangian dynamics from images for prediction and control". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 10741–10752 (↑ 35).

[288] Zwanzig, Robert. *Nonequilibrium statistical mechanics*. Oxford university press, 2001 (↑ 34).