Scuola Superiore Meridionale
University of Naples Federico II

Scuola Superiore Meridionale

Doctoral Thesis
in Modeling and Engineering Risk and Complexity

Data-driven modelling, analysis and control of crowd dynamics

Author:
Hector Vargas Alvarez

Principal Supervisor:
Prof. Constantinos Siettos, UNINA
Co:Supervisors:
Dr. Lucia Russo, STEMS,CNR
Prof. Jens Starke, Rostock University

Submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Modeling and Engineering Risk and Complexity. Coordinator: Prof. Mario di Bernardo.

Modelling and Engineering Risk and Complexity

26 April 2026

To my beloved parents, whose love, sacrifices, unwavering encouragement, and incessant support have been a source of strength throughout my life.

To my beloved fiancée, whose endless encouragement, patience, understanding, and love have helped me persevere throughout my PhD studies.

To all my family, who have always cared for me and my well-being, and who remained close despite the distance.

To you all, I dedicate the years of effort, perseverance, and learning that culminated in this work, with all my love and gratitude.

A mis queridos padres, cuyo amor, sacrificios, apoyo incondicional y constante aliento han sido una fuente de fortaleza a lo largo de toda mi vida.

A mi amada prometida, cuyo interminable aliento, paciencia, comprensión y amor me han ayudado a perseverar durante mis estudios de doctorado.

A toda mi familia, por su cariño, apoyo y preocupación constante por mi bienestar, y por mantenerse siempre cerca de mí a pesar de la distancia.

A ustedes, dedico los años de esfuerzo, perseverancia y aprendizaje que culminaron en este trabajo, con todo mi amor y gratitud.

Abstract

Complex systems exhibit emergent macroscopic behaviors arising from nonlinear interactions among large populations of agents. In application domains such as crowd dynamics, a fundamental open problem is to systematically bridge the gap between microscopic agent-based descriptions and predictive macroscopic models without relying on explicitly derived governing equations. This is essential for interpretable modeling and effective system-level numerical analysis and control tasks. The high dimensionality, heterogeneity, and nonlinearity of these systems make this micro–macro connection particularly challenging. In order to address this problem, this thesis work aims to develop a data-driven framework based on scientific machine learning and numerical analysis methods for learning macroscopic evolution operators in crowd dynamics directly from microscopic agent-based models, providing a structured foundation for system-level tasks such as state estimation and control.

The central premise of this thesis is that, although microscopic agent-based dynamics evolve in very high-dimensional spaces, the associated macroscopic behavior evolves on a low-dimensional manifold embedded in a high-dimensional function space. Macroscopic observables are constructed through the systematic coarse-graining of microscopic states and subsequent numerical analysis-based manifold learning, resulting in a reduced representation that preserves essential physical properties, such as mass conservation. This reduced representation provides the foundation for identifying macroscopic evolution operators governing the emergent crowd behavior.

Building on this reduced representation, the thesis formulates macroscopic crowd dynamics as the problem of identifying discrete-time evolution operators acting on the reduced macroscopic state space. In contrast to classical equation-free approaches based on local coarse time-steppers, the proposed framework identifies global macroscopic dynamics directly from data, without postulating explicit continuum equations. This operator-based formulation enables macroscopic simulations and long-horizon prediction of collective crowd behavior.

Numerical results demonstrate accurate recursive prediction and stable long-horizon behavior across different crowd interaction scenarios, while maintaining computational

efficiency and preserving key physical properties, such as mass conservation, by construction.

The reduced evolution operators identified for crowd dynamics enable the construction of controllers and observers directly within the same macroscopic state space, thereby establishing a unified formulation that connects model identification and state estimation in an end-to-end framework. A Physics-Informed Neural Network–based scheme is employed to enforce physical consistency in the design of controllers and observers. The resulting estimation framework is first validated on benchmark dynamical systems to assess its performance and is subsequently applied to the learned crowd operators, demonstrating accurate reconstruction of macroscopic states from partial observations. In addition, a control-oriented extension is developed within the same reduced-order setting and validated on benchmark dynamical systems. The integration of the complete control architecture with the crowd-dynamics framework developed in this thesis constitutes a direction for future work.

Acknowledgements

The path that every PhD student travels is inevitably filled with uncertainty, self-doubt, obstacles, and challenges. These experiences are a double-edged sword. On one hand, they make us question ourselves and our abilities; on the other, they offer the unique excitement of exploring the unknown, solving difficult problems, and embracing challenges that once seemed beyond our reach. It is precisely this balance between struggle and discovery that makes the doctoral journey both demanding and deeply rewarding.

Throughout this journey, I have been fortunate to encounter people who have played a fundamental role in my personal and scientific development. Thus, it is my will to express my deep gratefulness to them all. Firstly, I would like to express my deepest gratitude to Professor Constantinos Siettos. His scientific vision, guidance, and mentorship have shaped my way of thinking as a researcher. I am especially grateful for his constant availability, even during weekends and holidays, and his remarkable patience throughout the years. Above all, I thank him for teaching me, through his own example, what it means to be a critical and rigorous scientist.

Equally important, I would like to express my sincere gratitude to Professor Mario di Bernardo for his invaluable support since my arrival in Italy. I am deeply grateful for the constant attention he devoted to both my academic progress and personal well-being throughout my PhD journey. I would especially like to thank him for the time, guidance, and constructive feedback he provided, particularly during the early years of my doctoral studies.

His mentorship helped me develop skills that are not always explicitly taught in academia, including the importance of clear and effective scientific writing, the ability to communicate ideas convincingly, the value of delivering engaging presentations, and the importance of engaging with researchers from diverse backgrounds to broaden one's scientific perspective. These lessons have had a profound impact on both my professional and personal growth.

I would also like to express my sincere gratitude to Professor Jens Starke for his invaluable support during my time in Rostock. I am deeply grateful for his generosity with his time,

his willingness to discuss ideas and provide insightful feedback, and for the opportunity to collaborate with him. His guidance, encouragement, and scientific perspective greatly enriched my research experience and contributed significantly to my development as a researcher.

I would also like to express my sincere gratitude to Dr. Dimitrios Patsatzis, whose friendship, mentorship, and continuous support have been invaluable throughout my academic journey. I am equally grateful for the opportunity to collaborate scientifically with him. Furthermore, I would like to extend my sincere thanks to Dr. Gianluca Fabiani, Professor Nikolaos Kazantzis, and Professor Yannis Kevrekidis for their collaboration, insightful discussions, and teamwork, which led to important scientific publications. Working with them has been both a privilege and a valuable learning experience.

Importantly, I would like to express my deepest gratitude to Professor Alexander Poznyak and Professor Jesús Alberto Meda. At a crucial moment in my life, when I needed an opportunity, they did not hesitate to place their trust in me and offer their support. I am profoundly grateful for their generosity, invaluable help, and confidence in my potential.

Beyond their support, their scientific vision and dedication have been a lasting source of inspiration and have greatly influenced my academic and professional development. For all of this, I extend my heartfelt thanks.

Finally, I would like to express my deepest gratitude to my parents for their unconditional love, unwavering trust, and constant support in every path I have chosen to pursue, especially during moments of difficulty and setbacks. Their confidence in me has been a source of strength throughout my life.

Last but certainly not least, it is my wish to thank my beloved fiancée for her trust, patience, love, and following me to this extraordinary and often unpredictable journey. Her encouragement and companionship have meant more than words can adequately express. Please share this achievement with me, for without you this journey would not have been the same.

Contents

Part I

Foundations and Problem Setup

1 Introduction

Abstract This chapter establishes the scientific context and motivation of the thesis, by reviewing the state of the art and open problems in complex crowd dynamics and their broader scientific relevance. It examines the multiscale challenges inherent in modeling such systems and highlights the need for systematic micro–macro formulations. On this basis, the chapter formulates the central research questions and objectives guiding the proposed framework and concludes by summarizing its contributions and their importance in the context of risk and complexity.

1.1 Complex systems and emergent multiscale behavior

1.1.1 Emergence and nonlinearity in multi-agent systems

Complex systems consist of large populations of interacting agents (components) whose collective behavior gives rise to macroscopic patterns that are not explicitly encoded in the dynamics of the individual agents [1, 2, 3]. Such systems arise across physics, biology, engineering, and the social sciences, and are characterized by nonlinear interactions, feedback mechanisms, and adaptive responses [4, 5]. Unlike merely complicated systems, whose global behavior can in principle be reconstructed from a detailed decomposition of their parts, complex systems exhibit emergent phenomena: coherent large-scale structures arise spontaneously from local interaction rules without centralized coordination [6, 7, 8].

A defining feature of complex systems is nonlinearity. Small variations in local interactions may generate disproportionate or qualitatively distinct global outcomes, including synchronization, phase transitions, self-organization, or congestion patterns [9, 10]. These phenomena reflect the collective effects of many interacting degrees of freedom evolving simultaneously across space and time. As the number of agents increases, the dimensionality of the state space grows rapidly as well, thus making direct analytical treatment or exhaustive numerical exploration prohibitively costly or impractical [11, 12].

In addition to high dimensionality, complex systems inherently involve multiscale interactions: microscopic rules govern individual behavior at fine spatial and temporal scales, while macroscopic observables, such as density, momentum or flow, describe collective dynamics at coarser scales [13, 14]. Establishing the relationship between these levels of description, however, is neither trivial nor purely aggregative, as macroscopic quantities and their temporal evolution often emerge from intricate combinations of local interactions, and the governing principles at the collective scale are not directly deducible from microscopic rules alone.

As a result, this discrepancy between microscopic description and macroscopic behavior constitutes the micro–macro gap: a structural separation between the level at which dynamics are specified and the level at which prediction, interpretation, and control are typically required [11]. Understanding and formalizing this connection is central to the modeling of complex systems.

1.1.2 Modeling Across Scales in Crowd Dynamics

Human crowds represent a particular class of complex systems, as collective behavior arises from the interactions of many individuals operating under heterogeneous motivations, perceptions, and constraints. These systems naturally exhibit multiscale structure, where individual-level decisions and interactions give rise to emergent macroscopic patterns such as density profiles, lane formation, or congestion. Classical modeling formulations are typically organized into three distinct but interconnected levels of description:

the microscopic (individual), mesoscopic (kinetic), and macroscopic (hydrodynamic) scales. Each level provides complementary insight into collective dynamics, yet each also exhibits intrinsic limitations that complicate the establishment of a consistent and flexible micro–macro connection.

At the microscopic level, models describe the detailed behavior of individual pedestrians through agent-based approaches. Prominent examples include the Social Force Model (SFM) [15, 16] and its variants (see, e.g., [17, 18]), cellular automata formulations [19, 20, 21], and related rule-based or perceptual–cognitive interaction models. Another example in this category is the Visual Model, which assumes that pedestrians make movement decisions primarily based on information available within their field of view. Individuals continuously scan their surroundings and adjust their walking direction toward regions that appear less crowded and more navigable [22]. These approaches capture local interactions, behavioral heterogeneity, and decision-making mechanisms with high fidelity. However, the dimensionality of the resulting dynamical system scales with the number of agents, rendering large-scale simulation, systematic analysis, and control increasingly demanding. Although microscopic simulators may reproduce realistic emergent phenomena, extracting reduced macroscopic descriptions directly from high-dimensional agent-based dynamics remains nontrivial.

The mesoscopic scale provides an intermediate statistical description between individual-based and continuum formulations [23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. At this level, the system is characterized through distribution functions evolving in phase space, encoding probabilistic information about positions, velocities, and possibly internal states. Kinetic models are typically derived from microscopic interaction rules using tools from statistical mechanics. Through asymptotic analysis and moment methods, these formulations facilitate the derivation of macroscopic balance laws [26, 32]. In this sense, mesoscopic descriptions serve as a principled bridge between agent-level interactions and continuum-scale dynamics. Nevertheless, their derivation often relies on structural assumptions regarding interaction kernels, homogeneity, or population properties that may limit generality.

At the macroscopic scale, crowd behavior is described in terms of aggregated observables such as density, mean velocity, and flux, leading to continuum formulations based on conservation laws and suitable constitutive relations. Early approaches were inspired by fluid-dynamic analogies, giving rise to first-order models of Lighthill–Whitham–Richards type, where the flux is determined through a fundamental diagram relating density and velocity [33, 34]. To capture inertia and relaxation effects, second-order models were subsequently introduced, allowing for richer dynamical features such as stop-and-go waves and density instabilities [35, 36]. A seminal contribution specific to pedestrian dynamics is the model proposed by Hughes [37], which couples a conservation law with a nonlinear Eikonal equation to describe route-choice behavior via a potential field, thereby embedding anisotropy and decision-making mechanisms within a continuum framework. Extensions based on nonlocal conservation laws, in which fluxes depend on spatially averaged density, were developed in [38], capturing spatial anticipa-

tion effects. Complementarily, kinetic-theory-based frameworks provide a mesoscopic description of binary interactions and allow systematic derivations of macroscopic equations through moment-closure procedures [27]. More recent developments emphasize multiscale coupling strategies, behavioral heterogeneity, anisotropic interaction kernels, and perception-driven closures as emerging directions in macroscopic crowd modeling [32].

When high-fidelity microscopic simulators or detailed experimental data are available, a central objective is to derive mesoscopic or macroscopic models suitable for numerical analysis, optimization, and control of emergent collective behavior, for instance, in evacuation planning, infrastructure design, and flow regulation [25, 27, 39, 40, 41, 42, 43, 44]. In practice, the transition from high-dimensional agent-based representations to hydrodynamic descriptions is commonly achieved through closure techniques. These methods establish relationships between higher-order moments of microscopic distributions and a smaller set of lower-order macroscopic variables that capture observable emergent behavior [45, 46, 47].

However, such closure procedures inherently rely on modeling assumptions that may introduce structural biases [48, 49]. Typical assumptions include infinitely large populations underlying mean-field limits, homogeneous agents, and uniform or purely local interaction structures. While these approximations enable analytical tractability and rigorous derivations [50], they may significantly influence predictive accuracy and limit applicability in heterogeneous or data-rich environments. Moreover, the derivation of macroscopic equations through analytical limits is frequently model-specific, with closures depending strongly on the assumed interaction laws and scaling regimes [48, 49], making generalization across different crowd configurations challenging.

Thus, a central challenge in modeling human crowd dynamics is the systematic bridging of modeling scales. Although rigorous multiscale pathways exist in principle, the construction of reduced macroscopic descriptions that faithfully reflect the underlying microscopic dynamics remains a challenging and only partially resolved problem. Establishing systematic, flexible, and data-compatible methodologies for linking these scales, while preserving interpretability, physical consistency, and computational tractability therefore remains a central objective in the modeling of complex crowd systems.

1.2 Scientific Machine Learning/Data-driven approaches

The structural challenges identified in classical multiscale modeling, namely the high dimensionality of agent-based systems, the reliance on closure assumptions in macroscopic derivations, and the difficulty of systematically bridging microscopic and macroscopic descriptions have motivated the exploration of alternative modeling strategies. In settings where explicit macroscopic equations are unavailable, analytically intractable, or difficult to justify under heterogeneous and finite-size conditions, purely analytical scale-bridging approaches may become insufficient. These limitations have stimulated growing inter-

est in data-driven methodologies that exploit simulation data and observations to infer reduced dynamical representations directly.

In this context, machine learning and statistical inference provide flexible tools for extracting dynamical structure, approximating nonlinear mappings, and identifying reduced representations from high-dimensional data without requiring explicit analytical derivations. This paradigm has given rise to *scientific machine learning*, an interdisciplinary framework that integrates data-driven inference with established principles from physics, applied mathematics, and even control theory [51, 52, 53, 54].

Such methods offer expressive capabilities for approximating nonlinear dynamical operators that are analytically inaccessible. Neural architectures, including feed-forward networks [55, 56], recurrent models such as Long Short-Term Memory (LSTM) networks [57, 58, 59], deep neural networks (DNNs) [60], Generative Adversarial Networks (GANs) [61, 62], Convolutional Neural Networks (CNNs) [63, 64, 65], and reinforcement learning methods [66, 67], have been widely employed for modeling pedestrian, crowd, and traffic dynamics. Comprehensive reviews of these approaches are provided in [68]. These models have demonstrated strong performance in trajectory forecasting, density estimation, and pattern recognition, often operating without explicitly imposed physical structure.

In scientific applications, hybrid formulations such as Physics-Informed Neural Networks (PINNs) incorporate differential constraints or conservation laws directly into the learning process [53]. From a multiscale perspective, machine learning thus provides an alternative pathway for constructing surrogate models, identifying dominant collective variables, and approximating macroscopic evolution operators directly from microscopic simulations or observational data.

Despite these many advantages, purely data-driven strategies do not automatically resolve the structural challenges of multiscale modeling. Machine learning (surrogate) models typically require large volumes of high-quality data to be trained, they may lack interpretability, and do not inherently guarantee the preservation of conservation laws, symmetries, or stability properties. Predictive performance may deteriorate outside the training regime, limiting reliability in rare or extreme scenarios. Thus, these considerations underscore the necessity of structured, hybrid methodologies in which data-driven inference is embedded within mathematically grounded frameworks capable of preserving physical consistency while supporting analysis, estimation, and control.

Within this perspective, machine learning does not replace modeling; rather, it complements it by enabling the identification of macroscopic descriptions and dynamical operators directly from data. However, existing approaches differ substantially in how they incorporate physical structure, represent dynamics, and bridge modeling scales. It is therefore instructive to distinguish the principal methodological paradigms that characterize contemporary data-driven multiscale modeling.

1.2.1 Methodological paradigms in data-driven modeling

Recent developments in data-driven modeling of complex systems can be broadly categorized into three methodological paradigms.

The first paradigm concerns *equation discovery* and sparse system identification, where parsimonious governing laws are inferred directly from data. Techniques such as sparse regression and SINDy-type approaches aim to recover low-dimensional dynamical structures from time-series measurements [51, 69]. Extensions address partial differential equations and delay systems [70], enabling data-driven identification of governing equations in both ordinary and distributed-parameter settings.

The second paradigm encompasses *physics-informed learning*. Physics-Informed Neural Networks (PINNs) incorporate governing equations into the loss function, thereby enforcing differential constraints during training [53, 52]. Extensions include physics-informed neural operators [71, 72], and hybrid "grey-box" PDE identification strategies [45, 47, 72, 73, 74]. These approaches combine machine learning flexibility with physical priors such as conservation laws [75] and closure relations [76]. In crowd dynamics, PINNs have been employed to infer hydrodynamic PDE models [77] and to discover microscopic interaction rules [78]. Recent works further address stability and long-time accuracy [79, 80].

The third paradigm involves *operator-learning* methods, which approximate mappings between function spaces rather than individual trajectories. Neural operators, including DeepONet and Fourier Neural Operators, learn mappings from inputs such as initial conditions or parameters to solution fields [81, 82]. These approaches have been applied to accelerate PDE simulations and construct surrogate models for complex systems [83, 84]. In crowd modeling, they offer the possibility of approximating continuum-scale dynamics without explicitly deriving closed-form equations.

Collectively, these paradigms represent a shift from parameter estimation within predefined models toward direct identification of dynamical relationships and evolution operators from data.

1.2.2 Forward and inverse problems

Within data-driven modeling, it is common to distinguish between forward and inverse problem settings. A *forward* problem concerns predicting system outputs given a model, inputs, and parameters, whereas an *inverse* problem seeks to infer unknown parameters, latent states, or governing structures from partial and noisy observations [85, 86]. While forward problems are typically well-posed, inverse problems are often ill-posed and require regularization or prior information to ensure stability and identifiability [85].

In crowd dynamics, forward tasks include trajectory forecasting, density and flow prediction, and continuum-scale simulation. Deep sequence models such as Social LSTM and its extensions [58, 87, 88] have been widely adopted for multi-agent trajectory pre-

diction. At the macroscopic level, convolutional architectures dominate crowd-density estimation and counting from video data [89, 90, 91]. Hybrid approaches combine learned components with classical continuum solvers, including Hughes-type models [92, 93], while physics-informed and operator-learning surrogates have been proposed for real-time forecasting [53, 81, 82].

Inverse tasks in crowd modeling encompass parameter calibration, state estimation, and identification of latent interaction mechanisms. Bayesian and likelihood-based methods have been used to infer microscopic interaction parameters [94, 95, 96]. Sparse regression and equation-discovery techniques extend identification to PDEs and delay systems [51, 69, 70]. Hybrid physics–ML strategies have also enhanced kinetic models through data-driven identification of behavioral terms [97].

Although this forward/inverse classification clarifies the diversity of tasks addressed by data-driven approaches, it does not by itself resolve the structural multiscale issues identified earlier.

1.2.3 Structural gaps in data-driven approaches

Despite significant advances, several structural limitations persist. Physics-free machine learning models, while effective for short-term forecasting, lack interpretability and do not incorporate information about global spatio-temporal emergent quantities such as density fields that are central to macroscopic crowd modeling [68].

The equation-free multiscale framework [98] offers an alternative strategy by coupling microscopic simulations with coarse-graining and lifting operators to construct coarse time-steppers [99, 100, 101, 102, 103]. While powerful for numerical continuation and control tasks, this approach relies on locally constructed discrete maps rather than globally defined macroscopic evolution laws, limiting interpretability and long-term generalization.

Hybrid PINN-based approaches aim to learn PDE models from data [104, 71, 72, 45, 47, 72, 73, 74], incorporating conservation principles [75] and closures [76]. However, conservation laws, such as mass preservation, are typically enforced through soft constraints in the loss function [75, 105, 106], rather than guaranteed by construction. This indirect enforcement may limit robustness and reliability in heterogeneous or unforeseen scenarios.

Black-box neural operators and deep neural networks for learning conservation-law PDEs [47, 106, 107, 108, 109, 110] similarly lack inherent guarantees of physical consistency. Approximation errors and curse-of-dimensionality effects may lead to unphysical behavior or numerical instabilities during long-term simulations. Recent developments such as RiemannONets [111] and GoRINNs [112] attempt to enforce conservation laws by coupling neural networks with Riemann solvers or Godunov schemes, highlighting the importance of structural consistency.

Finally, it is worth noting that classical hydrodynamic PDE models for crowds rely on infinite-population assumptions. In realistic finite-size settings, spatial heterogeneity, boundary interactions, and localized behavioral variability may dominate, limiting the quantitative accuracy of continuum models.

Taken together, these observations indicate that a data-driven, end-to-end multiscale framework capable of modeling the evolution of macroscopic crowd dynamics, while ensuring physical consistency by construction, computational tractability, interpretability, and compatibility with system-level tasks such as state estimation and control, remains an open challenge.

The convergence of multiscale complexity, limitations of analytical closure, and structural weaknesses of existing data-driven approaches underscores the need for modeling frameworks that (i) identify reduced macroscopic representations directly from microscopic dynamics, (ii) enforce physical properties by construction, and (iii) remain computationally tractable and compatible with system-level analysis and control.

1.3 Research Objectives

The main goal of this thesis work is to advance the modeling, analysis, and control of emergent behavior in complex multi-agent systems, with particular emphasis on crowd and population dynamics. To this end, the thesis develops a data-driven framework that systematically bridges microscopic agent-based descriptions and macroscopic collective dynamics through reduced-order representations and identified evolution operators and their integration with methodologies for performing system-level tasks such as state estimation and control.

Specifically, the main objectives of this work are to:

1. **Establish a consistent micro–macro modeling pipeline** in which high-fidelity microscopic agent-based simulations serve as data-generating mechanisms for the identification of coarse-grained macroscopic dynamics.

2. **Identify appropriate macroscopic observables and their reduced state representations** governing collective behavior, using data-driven dimensionality-reduction and manifold-learning techniques to uncover low-dimensional structure embedded in high-dimensional microscopic data.

3. **Construct physically consistent micro–macro mappings** that connect agent-based dynamics to reduced macroscopic states, ensuring preservation of key physical properties such as mass conservation.

4. **Identify macroscopic evolution operators directly from data**, resulting in reduced-order dynamical models that describe the temporal evolution of coarse-grained states without assuming explicit macroscopic governing equations.

5. **Enable system-level tasks: state observation and control** by designing estimation and regulation strategies operating on the learned macroscopic models, thereby supporting prediction, reconstruction of latent states, and intervention at the collective level.

Together, these objectives define a coherent methodological pathway that connects microscopic simulation, coarse-grained representation, operator identification, and system-level tasks within an end-to-end framework. The resulting approach aims to provide both theoretical insight into multiscale dynamics and practical tools for forecasting, analysis, and control of collective behavior.

1.4 Key Research Questions

Building on the objectives outlined above, this thesis work addresses the following research questions:

1. **How can reduced and interpretable macroscopic models of collective behavior be constructed directly from microscopic simulations without explicitly deriving continuum equations?**

2. **Which macroscopic observables and reduced state variables capture the essential large-scale dynamics of crowd and population systems?**

3. **How can micro–macro mappings be identified in a data-driven yet physically consistent manner, ensuring preservation of structural physical properties such as mass conservation?**

4. **How can reduced-order evolution operators be learned to accurately reproduce and predict the temporal evolution of macroscopic quantities?**

5. **How can physically consistent, computationally efficient modeling architectures be designed to support reliable long-term prediction?**

6. **How can learned macroscopic models be integrated with estimation and control strategies?**

1.5 Contributions of this work

This thesis develops a data-driven framework based on machine learning and numerical analysis methods for manifold learning for the bridging of scales and the reduced-order modeling and control of complex multi-agent systems, with a particular emphasis on crowd dynamics. The central contribution does not lie in the isolated application of individual learning or reduction techniques, but in their systematic integration into a

coherent multiscale pipeline that connects microscopic agent-based simulations to physically consistent macroscopic representations and incorporates dedicated methodologies for system-level tasks such as state estimation and control.

The contributions of the thesis can be organized into two complementary directions: (i) the identification of macroscopic evolution operators from microscopic data via numerical analysis-based manifold learning and machine learning modelling methods, and (ii) the extension of these surrogate Reduced-Order models toward system-level observation and control. A visual overview of the framework proposed in this thesis work is depicted in Figure 1.1.

1.5.1 Reduced-Order Modelling of Macroscopic Crowd Dynamics

The first major contribution concerns the formulation of a data-driven methodology for identifying reduced-order macroscopic crowd dynamics directly from high-fidelity microscopic simulations. Inspired by the equation-free multiscale framework [98] and building on recent developments in next-generation equation-free algorithms based on manifold learning and machine learning [47, 101, 72, 113, 114, 115], the proposed framework identifies discrete macroscopic evolution operators in latent spaces without explicitly deriving continuum equations.

The methodology proceeds through four integrated stages. First, discrete microscopic agent distributions are mapped to continuous macroscopic density fields through a consistent coarse-graining procedure. Second, these macroscopic fields are projected onto a low-dimensional latent space using Proper Orthogonal Decomposition (POD), yielding reduced representations that capture the dominant collective modes of the system. Third, discrete-time reduced-order evolution operators are identified in the latent space using autoregressive models, including linear Multivariate Autoregressive (MVAR) models and nonlinear Long Short-Term Memory (LSTM) networks. The inclusion of temporal lags in these autoregressive formulations implicitly implements a delay-coordinate embedding, consistent with Takens' embedding theorem [116], thereby enabling reconstruction of the effective phase-space dynamics in the latent representation. Finally, the learned latent dynamics are lifted back to the high-dimensional macroscopic space via the POD basis, enabling reconstruction of full density fields.

A central design principle of the proposed framework is the enforcement of physical consistency by construction. In particular, we have proven that the POD representation preserves the mass of the macroscopic density field, which is enforced explicitly through the micro–macro mapping and reduced representation rather than imposed weakly via soft penalty terms in learning objectives. By comparing linear and nonlinear reduced-order operators, the thesis demonstrates that accurate and generalizable macroscopic dynamics can be captured using parsimonious and interpretable models, mitigating the curse of dimensionality while maintaining computational efficiency.

The methodology is validated using high-fidelity simulations of pedestrian dynamics

based on the Social Force Model (SFM) [15, 16], considering both unidirectional and dense counterflow scenarios in a corridor with obstacles under periodic boundary conditions. The learned operators are evaluated on unseen initial conditions, demonstrating predictive capability and robustness. This contribution establishes a structurally consistent operator-based modeling layer that systematically bridges microscopic agent interactions and macroscopic collective behavior within a unified equation-free framework.

1.5.2 State estimation and control

Complementing the data-driven multiscale modeling framework, the second contribution of the thesis addresses system-level tasks, particularly state estimation and control. These methodologies are developed in a general dynamical-systems setting and are designed to operate on reduced-order or data-driven models, including those obtained through the multiscale operator-identification pipeline introduced above.

A first contribution in this direction concerns nonlinear state estimation formulated within a physics-informed learning observer-design framework. Specifically, nonlinear coordinate transformations required for observer design are computed numerically using Physics-Informed Neural Networks, embedding the structural constraints of the observer equations directly into the learning process. This provides a flexible alternative to classical analytical or series-based observer construction, particularly in regimes where closed-form transformations are unavailable.

The observer methodology is validated on benchmark nonlinear systems and *subsequently integrated with the learned macroscopic evolution operators arising from the crowd dynamics modeling framework*, thereby completing the end-to-end framework depicted in Figure 1.1. In this setting, when macroscopic dynamics are represented by nonlinear reduced-order time-steppers in latent space, observer design is performed directly in the learned reduced coordinates. When linear reduced-order operators are available, classical estimation strategies can be employed. This integration demonstrates how structurally consistent, reduced-order macroscopic models enable state reconstruction under partial observability without reverting to high-dimensional microscopic simulations.

In parallel, a feedback linearization control-oriented framework is developed and tested on representative nonlinear benchmark problems. The proposed control strategies are formulated to operate on reduced-order dynamical models and are designed to preserve structural properties of the underlying system. While the control framework is not yet fully integrated with the crowd dynamics application, its formulation is directly compatible with the operator-based multiscale modeling pipeline, and the thesis establishes the conceptual and methodological foundations required for such integration.

Together, the modeling, estimation, and control contributions establish a coherent architecture that bridges microscopic simulation, reduced macroscopic representation, operator identification, and system-level tasks. The resulting framework provides physically consistent, computationally efficient, and interpretable models suitable for prediction,

state estimation, and control of collective dynamics. Although motivated and illustrated primarily through crowd dynamics, the methodology is, in principle, applicable to other complex multi-agent systems characterized by multiscale structure, nonlinear interactions, and partial observability.

1.6 Relevance to complexity

Crowd systems are prototypical complex systems, composed of many interacting agents whose local decisions, heterogeneity, and stochasticity collectively generate macroscopic patterns that are nonlinear, history-dependent, and often difficult to predict [117, 118]. These emergent phenomena are central to a wide range of high-impact societal applications, including the safe evacuation of public spaces during emergencies, the management of pedestrian and vehicular mobility in dense urban environments, and the design of resilient and sustainable infrastructures. In such contexts, the inability to anticipate collective behavior can lead to cascading failures, severe safety hazards, and systemic risks [119, 120].

A central unresolved challenge in this domain is the systematic connection between microscopic and macroscopic descriptions of collective dynamics. Microscopic models such as social-force and agent-based formulations capture individual interactions and behavioral variability, but become computationally expensive and difficult to analyze at scale [15]. Conversely, macroscopic and continuum models offer analytical tractability and computational efficiency, yet rely on phenomenological closures that are difficult to derive rigorously from first principles [92]. This scale gap limits the development of predictive and controllable models suitable for risk analysis, forecasting, and real-time intervention.

The methodology developed in this thesis directly addresses these challenges by proposing a unified, data-driven, and control-oriented framework for multiscale modeling of collective dynamics. By integrating numerical analysis, manifold learning, and control theory, the proposed approach enables the extraction of low-dimensional macroscopic representations from high-dimensional microscopic data while retaining essential behavioral information [121]. This multiscale perspective provides a principled pathway to connect individual interactions with emergent collective behavior, enhancing interpretability, scalability, and robustness.

From the standpoint of complexity, this framework enables quantitative characterization, prediction, and regulation of emergent behaviors under uncertainty. Reduced-order yet physically and behaviorally consistent models support scenario analysis, early warning indicators of instability, and control strategies for mitigating systemic risks [122]. Beyond crowd dynamics, the concepts and methods developed in this work are broadly applicable to other complex socio-technical systems, positioning the thesis at the intersection of complexity science, data-driven modeling, and risk-aware decision-making for human-centered environments.

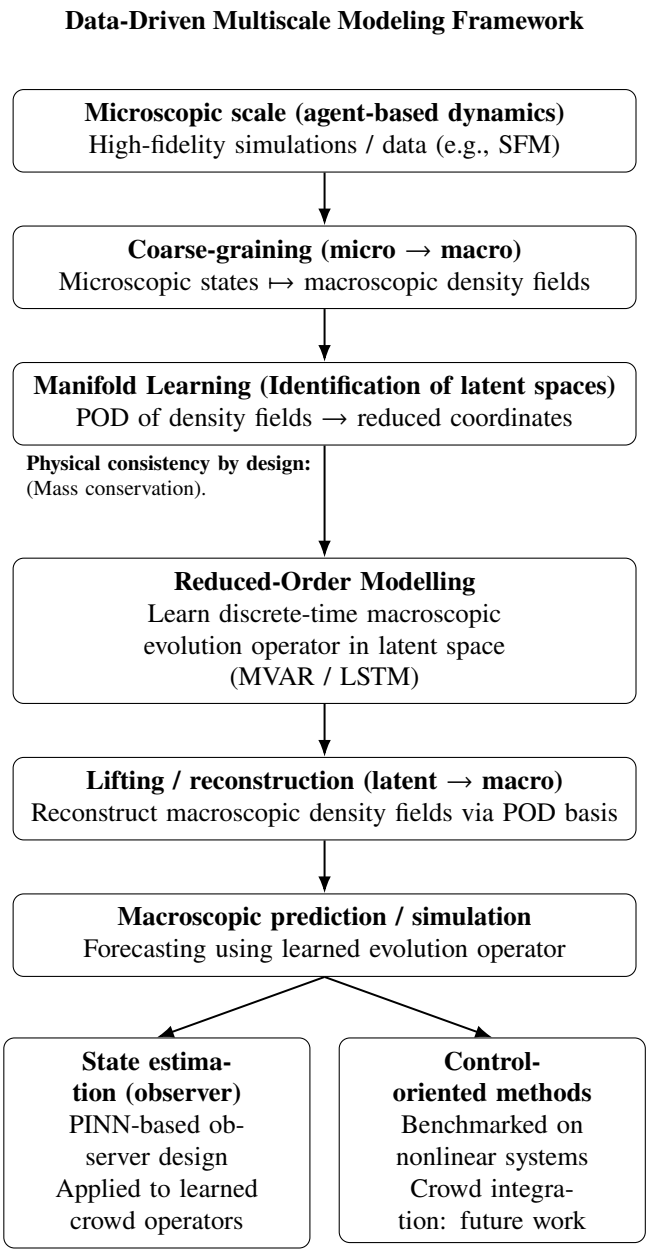


Figure 1.1: Data-driven multiscale framework: agent-based dynamics, coarse-graining, latent reduction, evolution operator identification, reconstruction, and macroscopic prediction, enabling state estimation and control-oriented tasks.

Publications from this thesis

Parts of this thesis are based on the following peer-reviewed and preprint publications authored by the candidate. The thesis substantially extends these works by providing a unified equation-free framework, consistent notation, additional theoretical context, and integrated applications to crowd dynamics, estimation, and control.

- H. V. Alvarez, D. G. Patsatzis, L. Russo, I. G. Kevrekidis, and C. Siettos, "Next generation equation-free multiscale modelling of crowd dynamics via machine learning," *Journal of Computational Physics*, vol. 559, 114881, 2026. [123]

  This work provides the foundation for the equation-free, multiscale modeling framework developed in Chapters 3-5. In particular, it introduces the operator-based identification of reduced macroscopic dynamics from microscopic agent-based simulations. The thesis extends this work by embedding it within a broader modeling pipeline, by emphasizing physical consistency (e.g. mass conservation by construction), and by integrating the learned macroscopic models with system-level estimation and control tasks.

- **[P2]** H. Vargas Alvarez, G. Fabiani, N. Kazantzis, I. G. Kevrekidis, and C. Siettos, *Nonlinear discrete-time observers with Physics-Informed Neural Networks*, Chaos, Solitons & Fractals, vol. 186, p. 115215, 2024.[124]

  This publication underpins the observer design methodologies presented in Chapter 6. While the original article develops physics-informed observers for general nonlinear discrete-time dynamical systems, the thesis integrates these observer formulations with the reduced-order macroscopic models learned from microscopic crowd simulations.

- **[P3]** H. Vargas Alvarez, G. Fabiani, N. Kazantzis, C. Siettos, and I. G. Kevrekidis, *Discrete-Time Nonlinear Feedback Linearization via Physics-Informed Machine Learning*, Journal of Computational Physics, vol. 492, p. 12408, 2023. [125]

  This publication contributes to the physics-informed control methodologies discussed in Chapter 7. The paper develops a PINN-based approach to compute nonlinear coordinate transformations enabling discrete-time feedback linearization. In the thesis, this approach is identified as a natural candidate for addressing control problems in crowd dynamics, and is discussed as a forward-looking component for future extensions of the proposed multiscale framework.

1.7 Thesis structure and outline

The thesis is organized into three main parts.

The first part establishes the scientific and methodological foundations of the providing a broader state-of-the-art on complex systems and multiscale modeling. Chapter 2 provides the necessary mathematical and methodological background, covering equation-free multiscale modeling, reduced-order techniques, and data-driven approaches for learning dynamical systems. Chapter 3 presents the microscopic modeling framework employed throughout the thesis, describing the agent-based crowd model, numerical simulation procedures, and the construction of macroscopic observables from particle-based representations.

The second part contains the core modeling contributions of the thesis. Chapter 4 develops a data-driven framework for modeling macroscopic crowd dynamics, focusing on the construction of physically consistent macroscopic state spaces and reduced representations for high-dimensional density fields. Macroscopic observables are interpreted as elements of a low-dimensional manifold embedded in a high-dimensional function space, and mass-conserving micro–macro mappings are introduced to enable reduced-order representations without relying on explicit continuum closures. Chapter 5 builds upon this state-space formulation and constitutes the principal dynamical contribution of the thesis: a data-driven framework for identifying reduced-order macroscopic evolution operators directly from microscopic simulations, enabling computationally efficient modeling and long-horizon prediction of collective behavior.

The final part of the thesis extends the learned macroscopic models to system-level tasks. Chapter 6 develops a physics-informed machine learning (PIML)–based observer framework for state estimation on the reduced macroscopic dynamics and integrates it with the learned crowd operators, thereby unifying the multiscale modeling architecture with system-level state reconstruction under partial observability. Finally, Chapter 7 presents a control-oriented methodology for discrete nonlinear systems, also based on PIML, validated on benchmark problems and formulated to be structurally compatible with the data-driven multiscale modeling framework developed in this thesis work.

2 Methodological Foundations

Abstract This chapter reviews the methodological foundations underlying the scientific machine learning approach developed in this thesis work. We revisit the classic equation-free paradigm for the analysis of multiscale systems, reduced-order modeling techniques for the representation of high-dimensional macroscopic states, and data-driven approaches for learning discrete-time evolution operators. Together, these concepts provide the theoretical and computational basis for the learning of macroscopic collective dynamics developed in the remainder of the thesis.

2.1 Multiscale Modeling and the micro–macro Gap

As mentioned in the introduction, complex systems are characterized by the coexistence of processes evolving across multiple spatial and temporal scales. Such *multiscale systems* typically consist of a large number of interacting agents (components) whose fast-scale, nonlinear dynamics collectively give rise to emergent slow-scale behavior observable at the macroscopic level. The resulting separation between microscopic interactions and macroscopic organization is a defining feature of multiscale systems.

This behavior is often associated with a separation of time scales, whereby fast microscopic dynamics rapidly relax toward a low-dimensional manifold that governs the slower macroscopic evolution. As a consequence, the long-term dynamics of the system can often be described in terms of a reduced set of coarse variables evolving on slower time scales.

Classical examples arise in fluid dynamics and collective motion. In fluid systems, microscopic particle interactions occur at very fast time scales, while macroscopic quantities such as velocity and pressure fields evolve more slowly and admit continuum descriptions. Similarly, in systems of interacting agents, individual-level decisions and interactions take place locally, yet produce aggregate phenomena such as density waves, lane formation. In both cases, microscopic descriptions are naturally formulated in high-dimensional state spaces, whereas macroscopic descriptions aim to capture dominant collective behavior through a reduced set of variables [126, 127].

Thus, a central challenge in multiscale modeling lies in establishing a systematic connection between microscopic and macroscopic descriptions. In many practical settings, macroscopic governing equations cannot be derived rigorously from microscopic dynamics due to strong nonlinearities, stochastic effects, complex interaction mechanisms, or the absence of a clear scale separation. As a consequence, classical analytical techniques based on averaging or homogenization may fail to provide closed or accurate macroscopic models. This fundamental difficulty is commonly referred to as the *micro–macro gap* [128].

These challenges motivate the development of computational and data-driven approaches that enable macroscopic analysis directly from microscopic simulations, without relying on explicit closed-form macroscopic equations. One prominent class of such methodologies is provided by the *equation-free* framework, which is introduced in the following subsection.

2.2 Equation-free modeling paradigm

Equation-free (EF) modeling is a computational paradigm designed to enable macroscopic analysis of complex multiscale systems in situations where explicit macroscopic governing equations are unavailable, impractical to derive, or insufficiently accurate [129, 121]. The central objective of the EF approach is to perform system-level tasks, such as simulation, fixed-point computation, stability and bifurcation analysis [100, 130], or control [131, 54], by exploiting detailed microscopic simulators directly, while operating as if an explicit macroscopic model were available.

In the equation-free framework microscopic models, such as agent-based or particle-based simulators, are treated as *black-box local time steppers* that encode the system's fine-scale dynamics. Rather than attempting to derive closed-form macroscopic equations, the EF methodology works under the key assumption that a suitable set of coarse or macroscopic variables exists and that it is capable of capturing the dominant collective behavior of the system. Short bursts of microscopic simulation are then used to estimate the temporal evolution of these coarse variables, thereby bypassing the need for explicit macroscopic equations [132, 133].

A key philosophical aspect of equation-free modeling is the separation between *representation* and *computation*. Macroscopic variables are introduced to represent the system at a coarse level, but their evolution is computed indirectly through controlled interactions with the microscopic simulator. In this sense, equation-free methods do not replace microscopic models; rather, they elevate them to serve as computational engines for macroscopic analysis. This perspective enables the reuse of existing high-fidelity simulators for macroscopic tasks that would traditionally require explicit continuum-scale models.

Equation-free methods are particularly well suited for multiscale systems characterized by strong nonlinearities, stochastic effects, and emergent collective behavior, where classical analytical approaches such as averaging, homogenization, or kinetic limits may fail or rely on restrictive assumptions. This is the case, in particular, for the macroscopic modeling of crowd dynamics, where collective behavior emerges from nonlinear, local interactions among a large number of agents.

2.2.1 Core components

We now describe the fundamental components of the equation-free framework and how they are combined to approximate macroscopic evolution in the absence of explicit governing equations.

Let

$$u(t) \in \mathbb{R}^M \tag{2.1}$$

denote the microscopic state of the system at time t, where M is typically large and corresponds to the degrees of freedom of the fine-scale model. The macroscopic or

coarse state is represented by a vector

$$U(t) \in \mathbb{R}^d, \qquad d \ll M, \tag{2.2}$$

which captures the essential system dynamics at the macroscopic level.

A fundamental modeling assumption underlying equation-free methods is that the evolution of $U(t)$ is approximately autonomous, in the sense that it is largely independent of unresolved microscopic details once appropriate coarse variables have been identified. This assumption reflects the existence of a low-dimensional slow manifold governing the emergent system behavior and plays a central role in the validity of the EF approach.

The identification of suitable coarse variables is therefore a critical and nontrivial aspect of equation-free modeling. While classical approaches often rely on physical intuition or analytical insight, the challenges posed by high-dimensional and data-rich systems motivate the use of systematic, data-driven techniques for discovering effective macroscopic representations, as discussed in later chapters of this thesis.

Restriction operator

Given a choice of coarse variables, the EF framework requires a systematic procedure for mapping microscopic system states to their macroscopic representations. This is accomplished through the so-called *restriction operator*, which is defined as follows

$$\mathcal{R} : \mathbb{R}^M \to \mathbb{R}^d, \qquad U(t) = \mathcal{R}(u(t)), \tag{2.3}$$

This operator d extracts macroscopic information from a microscopic realization. The choice of $\mathcal{R}$ is problem-dependent and reflects the physical or statistical quantities of interest at the macroscopic level. Common examples typically include spatial averages, low-order statistical moment or coarse-grained field representations that filter out microscopic fluctuations while retaining dominant collective features [130, 131].

In practice, the restriction operator is assumed to be straightforward to compute from microscopic simulators, which naturally provide access to detailed state information. However, $\mathcal{R}$ is generally not invertible: many distinct microscopic configurations may correspond to the same macroscopic state. This non-invertibility has important implications for the construction of macroscopic evolution and necessitates the introduction of a complementary lifting operator.

Lifting operator

While the restriction operator maps microscopic states to coarse variables, equation-free methodology requires a mechanism for initializing microscopic realizations (simulations) in consistent manner with a prescribed macroscopic state. This role is played by the *lifting operator*.

The lifting operator is defined as a mapping

$$\mathcal{L} : \mathbb{R}^d \to \mathbb{R}^M, \qquad u(t) = \mathcal{L}(U(t)), \tag{2.4}$$

which constructs a microscopic state consistent with a given macroscopic description. Because the restriction operator is generally non-invertible, the lifting map is not unique: a given macroscopic state typically corresponds to a set of admissible microscopic configurations.

In practice, lifting may be implemented through randomized reconstruction procedures, constrained sampling, or short equilibration simulations that allow fast microscopic transients to relax while preserving the prescribed coarse observables. The design of accurate lifting operators is often one of the most delicate aspects of EF modeling, as inconsistencies between lifting and restriction may introduce spurious microscopic artifacts that contaminate coarse-scale predictions. For this reason, careful validation of lifting–restriction consistency is essential for reliable EF computations [54].

Coarse time-stepper

We now describe how the restriction and lifting operators are combined with microscopic simulators to approximate the evolution of macroscopic variables.

Let

$$u(t + \Delta t) = \Phi_{\Delta t}(u(t)) \tag{2.5}$$

denote the microscopic time-stepper associated with the fine-scale simulator, where $\Phi_{\Delta t}$ advances the microscopic state over a time interval Δt. In the absence of an explicit macroscopic evolution equation, EF paradigm defines an implicit macroscopic time-stepper by composing lifting, microscopic evolution, and restriction. Therefore, given a macroscopic state $U(t)$, its coarse evolution over a time step Δt is approximated as

$$U(t + \Delta t) \approx \mathcal{R}(\Phi_{\Delta t}(\mathcal{L}(U(t)))) \,. \tag{2.6}$$

Equation (2.6) defines an implicit macroscopic evolution operator that advances the coarse variables without requiring an explicit closed-form macroscopic model. Under appropriate assumptions, such as the existence of a low-dimensional slow manifold and sufficient time-scale separation, the resulting coarse dynamics approximate the evolution that would be obtained from an explicit macroscopic equation, were such an equation available. In practice, short microscopic simulation intervals are often sufficient, as fast microscopic transients rapidly relax toward the slow macroscopic manifold.

The coarse time-stepper (2.6) forms the basis for a wide range of macroscopic computational tasks within the equation-free framework, including coarse projective integration, fixed-point computation, and stability or bifurcation analysis. In all cases, microscopic simulators are used only for short bursts (intervals) in time, while macroscopic computations are performed at the level of coarse variables.

In practical applications, EF method is affected by stochasticity and sensitivity in microscopic simulations, which may introduce noise into coarse-scale estimates and necessitate set averaging or variance-reduction strategies. Moreover, classical EF computations rely

on repeated short bursts of microscopic simulation to approximate macroscopic evolution, leading to increased computational cost for long-time integration or extensive parameter exploration. A further characteristic of traditional EF implementations is their inherently *local* nature, both in time and, in many cases, in the macroscopic state space.

These limitations motivate the integration of EF ideas together with reduced-order modeling and data-driven learning techniques, with the aim of constructing explicit global macroscopic evolution operators directly from microscopic simulations. Such approaches, which combine EF with learned reduced representations, form the methodological basis of the framework developed in the subsequent chapters of this thesis.

2.3 Reduced-order modeling and manifold learning

Even when macroscopic variables are available, their direct use in simulation, analysis, and control often remains computationally prohibitive, even with modern computational resources owing to the fact that macroscopic descriptions of multiscale systems frequently take the form of spatially distributed fields, such as densities or velocity fields, which upon discretization, give rise to high-dimensional dynamical systems. The dimension of these systems grows rapidly with spatial resolution, resulting in substantial computational and storage requirements [134, 135].

This rapid growth in computational complexity is commonly referred to as the *curse of dimensionality*, which in computational practice, it manifests as an exponential increase in the number of degrees of freedom required to accurately represent system states, together with a corresponding increase in the cost of time integration, parameter exploration, optimization, and control. As a consequence, even macroscopic models that are conceptually simpler than their microscopic counterparts may become impractical for long-time simulation or repeated evaluations, particularly in data-driven or control-oriented settings.

From a dynamical systems perspective, however, it is often observed that the effective behavior of such high-dimensional systems evolves on a much lower-dimensional structure embedded in the ambient state space. Despite the apparent complexity of the full macroscopic description, the dominant dynamics may be governed by a small number of modes that capture the essential (key) features of the system's evolution. This observation underlies many successful reduced modeling approaches in fluid dynamics, chemical kinetics, and other areas of applied mathematics, where long-term dynamics are well approximated by low-dimensional invariant or attracting sets [135, 136, 137].

Reduced-order modeling (ROM) seeks to exploit this structure by constructing low-dimensional representations of high-dimensional system states. This is, typically, achieved by projecting the original dynamics onto a reduced space spanned by a small number of basis functions or modes that preserve the dominant behavior of the system and by working in this reduced space, one can substantially alleviate the computational burden

associated with high-dimensional macroscopic models while retaining sufficient accuracy for system-level tasks such as simulation, analysis, and control [138, 139, 140].

Therefore, it is important to emphasize that reduced-order modeling primarily addresses the problem of *representation*, rather than that of *dynamics*. That is, ROM techniques aim to identify compact coordinates in which macroscopic states can be efficiently described, but they do not, by themselves, provide explicit evolution laws for these reduced variables. This distinction is crucial in the context of this thesis work: reduced representations serve as a necessary precursor to the learning of macroscopic evolution operators, but they do not replace the need for dynamical modeling.

Within the EF framework, introduced in the previous section, reduced-order representations provide a natural interface between microscopic simulations and data-driven learning approaches. By restricting attention to a low-dimensional macroscopic state space, it becomes feasible to approximate, learn, and analyze evolution operators acting on reduced variables, thereby enabling the construction of explicit macroscopic dynamical models from microscopic data.

These considerations motivate the introduction of systematic techniques for constructing reduced-order representations of high-dimensional macroscopic systems. Among such techniques, Proper Orthogonal Decomposition (POD) [140, 139] provides a mathematically principled and widely used framework for identifying low-dimensional linear subspaces that optimally capture the variance of observed system states. POD serves both as a baseline linear reduction method and as a foundational tool for subsequent nonlinear manifold learning approaches. This method is reviewed in the following subsection.

2.3.1 Proper orthogonal decomposition

The central idea of POD is to identify a low-dimensional subspace in which a collection of observed macroscopic states can be approximated with minimal error in the least-squares (L^2) sense. To formalize this setting, let $\mathcal{H}$ denote a Hilbert space equipped with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\| \cdot \|_{\mathcal{H}}$, representing the space of macroscopic states. We consider a time-dependent macroscopic state $u(t) \in \mathcal{H}$ and assume access to a finite set of observation data

$$\{u(t_1), u(t_2), \ldots, u(t_{n_t})\} \subset \mathcal{H}. \tag{2.7}$$

The objective of POD is to construct a linear subspace $V_d \subset \mathcal{H}$ of dimension $d \ll M$ that provides an optimal approximation of the observation set in a least-squares sense.

Optimal low-dimensional subspaces

The POD approximation problem may be formulated as the following problem

$$\min_{\substack{V_d \subset \mathcal{H} \\ \dim(V_d)=d}} \frac{1}{n_t} \sum_{k=1}^{n_t} \left\| u(t_k) - P_{V_d} u(t_k) \right\|_{\mathcal{H}}^2, \tag{2.8}$$

where P_{V_d} denotes the orthogonal projector onto V_d. This formulation makes explicit that POD seeks the linear subspace of prescribed dimension that minimizes the average projection error over the observation data.

A fundamental result underlying POD is that the solution of (2.8) is given by the subspace spanned by the leading eigenfunctions of the empirical covariance operator associated with the observation set. This optimality property distinguishes POD from heuristic projection techniques and provides a clear criterion for assessing the quality of reduced representations [134, 141, 142].

The eigenvalues associated with the POD modes quantify the variance, or energy, captured by each mode and therefore provide a natural basis for selecting the truncation dimension d. Rapid decay of these eigenvalues indicates that the macroscopic states admit an efficient low-dimensional linear representation.

Reduced coordinates

Let $\{\psi_i\}_{i=1}^d \subset \mathcal{H}$ denote an orthonormal basis of the POD subspace V_d. A macroscopic state $u(t)$ can then be approximated by its orthogonal projection onto V_d,

$$u(t) \approx \sum_{i=1}^{d} a_i(t)\, \psi_i, \tag{2.9}$$

where the coefficients

$$a_i(t) = \langle u(t), \psi_i \rangle_{\mathcal{H}}, \qquad i = 1, \dots, d, \tag{2.10}$$

define the reduced coordinates of the system state in the POD basis. Introducing the vector of reduced coordinates

$$a(t) = (a_1(t), \dots, a_d(t))^\top \in \mathbb{R}^d, \tag{2.11}$$

the approximation (2.9) can be written compactly as

$$u(t) \approx \Psi a(t), \tag{2.12}$$

where $\Psi = [\psi_1, \dots, \psi_d]$ denotes the operator whose columns are the POD modes. This representation provides a compact parametrization of the macroscopic state. In practical computations, macroscopic states are represented through spatial discretization,

resulting in high-dimensional vectors in $\mathbb{R}^{n_c}$. Collecting n_t such observations leads to the observation matrix

$$U = [u_1, u_2, \ldots, u_{n_t}] \in \mathbb{R}^{n_c \times n_t}, \tag{2.13}$$

Here, n_c denotes the dimension of the discretized macroscopic state and n_t the number of temporal observations. whose columns correspond to discretized realizations of $u(t_k)$. In this finite-dimensional setting, the POD modes can be computed efficiently via the singular value decomposition (SVD) of the observation matrix, or equivalently through an eigenvalue decomposition of the associated correlation matrix [134, 142]. This observation-based formulation provides a direct computational realization of the abstract POD construction. This separation between representation and dynamics is deliberate and aligns naturally with the EF paradigm introduced earlier. In particular, POD supplies a reduced macroscopic state space on which evolution operators can subsequently be identified or learned from data. The resulting reduced representation can then be used as a computationally tractable surrogate for the original high-dimensional macroscopic state.

Remark 2.3.1. While POD provides an optimal linear approximation of the observation set in a least-squares sense, its effectiveness depends critically on the structure of the underlying dynamics. In particular, slow decay of the POD eigenvalues may indicate that a larger number of modes is required to capture the relevant behavior, reducing the efficiency of linear reduction [113]. Moreover, POD is inherently limited to linear subspaces and may be insufficient for systems whose macroscopic dynamics evolve on strongly nonlinear or curved manifold.

2.4 Learning dynamical systems from data

The reduced-order representations introduced in the previous section provide a compact description of the system state. The remaining challenge is to model the temporal evolution of these reduced variables when explicit governing equations are unavailable. In data-driven settings, this task must be accomplished using only simulation or observational data. This section reviews approaches for learning dynamical systems directly from data.

2.4.1 Surrogate Model learning and discrete-time steppers

A useful viewpoint for dynamical systems is to represent time evolution through an *evolution operator* acting on an appropriate state space. This perspective is particularly natural in equation-free (EF) settings, where explicit governing equations may be unavailable but time advancement can still be observed through simulation or data [141, 143].

Let $\boldsymbol{y}(t)$ denote the system state at time t. For a continuous-time autonomous system

$$\dot{\boldsymbol{y}}(t) = f(\boldsymbol{y}(t)), \tag{2.14}$$

the solution defines a flow map Φ_t such that

$$\boldsymbol{y}(t) = \Phi_t(\boldsymbol{y}_0), \tag{2.15}$$

where $\boldsymbol{y}_0 = \boldsymbol{y}(0)$. In practice, system information is often available only at discrete time instants. Fixing a sampling interval $\delta t > 0$ and defining $t_k = k\delta t$, the sampled dynamics can be written as

$$\boldsymbol{y}(t_{k+1}) = \Phi_{\delta t}(\boldsymbol{y}(t_k)), \qquad \boldsymbol{y}(t_k) \approx \boldsymbol{y}(t_k). \tag{2.16}$$

Given data pairs $\{(\boldsymbol{y}(t_k), \boldsymbol{y}(t_{k+1}))\}_{k=0}^{n_t-1}$ obtained from measurements or simulation, operator learning seeks a parametric approximation $F_{\boldsymbol{\theta}}$ of the unknown evolution operator such that

$$F_{\boldsymbol{\theta}}(\boldsymbol{y}(t_k)) \approx \boldsymbol{y}(t_{k+1}). \tag{2.17}$$

Here $F_{\boldsymbol{\theta}}$ denotes a learned surrogate model, parameterized by $\boldsymbol{\theta}$, that approximates the action of the true evolution operator $\Phi_{\delta t}$. This formulation underlies many modern data-driven approaches to dynamical system modeling. Of particular interest are *global surrogate models* that approximate the evolution operator over a region of interest in the state space, enabling repeated and inexpensive evaluations without further access to the underlying simulator [143].

In the methodology developed in this thesis, the system state $\boldsymbol{y}(t_k)$ corresponds to reduced coordinates obtained through the dimensionality reduction procedures described in the previous section. Consequently, the learned operator approximates the temporal evolution of the system directly in the reduced state space.

2.4.2 Recurrent neural networks as discrete-time dynamical systems

Recurrent neural networks (RNNs) provide a flexible class of parameterized models for approximating discrete-time evolution operators from data. From a mathematical perspective, an RNN defines a nonlinear state-space system in which the observable state is augmented with hidden variables that encode information about past system behavior. Consequently, RNNs may be interpreted as evolution operators acting on an extended state space that includes both observable variables and internal hidden states. This formulation enables the model to capture temporal dependencies in the data while maintaining a discrete-time nonlinear dynamical system structure [144].

Let $\boldsymbol{y}(t_k) \in \mathbb{R}^d$ denote the observable state at discrete time t_k, and let $\mathbf{h}_{t_k} \in \mathbb{R}^p$ denote a hidden state. A standard RNN is defined by

$$\boldsymbol{h}_{k+1} = \sigma\left(\mathbf{W}_h \mathbf{h}_{t_k} + \mathbf{W}_y \boldsymbol{y}(t_k) + \mathbf{b}_h\right), \tag{2.18}$$

$$\hat{\boldsymbol{y}}_{k+1} = \mathbf{W}_y \boldsymbol{h}_{k+1} + \mathbf{b}_y, \tag{2.19}$$

where σ is a nonlinear activation function. The parameter matrices $\mathbf{W}_h \in \mathbb{R}^{p\times p}$, $\mathbf{W}_y \in \mathbb{R}^{p\times d}$, and $\mathbf{W}_y \in \mathbb{R}^{d\times p}$, together with the bias vectors $\mathbf{b}_h \in \mathbb{R}^p$ and $\mathbf{b}_y \in \mathbb{R}^d$, define the trainable parameters of the model.

This formulation induces an evolution operator on the augmented state space $\mathbb{R}^p \times \mathbb{R}^d$, where the hidden variables are treated as additional state components. Eliminating the hidden state yields an effective input–output operator of the form

$$\hat{\boldsymbol{y}}_{k+1} = F_{\boldsymbol{\theta}}(\boldsymbol{y}(t_k), \boldsymbol{y}_{k-1}, \dots). \tag{2.20}$$

where the dependence on past states reflects the finite memory encoded by the hidden dynamics. In this way, RNNs extend classical discrete-time evolution maps by incorporating temporal context directly into the state representation.

The hidden-state recursion defines a nonlinear discrete-time dynamical system whose stability properties depend on the recurrent weight matrix and the activation function. Under suitable contractivity conditions, the hidden dynamics are stable and exhibit a fading-memory property, whereby the influence of past states decays with time lag [144]. While such conditions ensure well-posedness and robustness, they also limit the ability of classical RNNs to represent long-term dependencies. This stability–memory tradeoff places intrinsic limitations on standard RNN architectures when modeling multiscale dynamical systems, in which slow macroscopic modes may require persistent memory over extended temporal horizons. As a result, classical RNNs often struggle to capture long-term dependencies without compromising stability or trainability.

The limitations of standard recurrent networks motivate the development of alternative architectures that explicitly regulate information flow and memory retention. In particular, gated recurrent models such as long short-term memory (LSTM) networks introduce additional state variables and multiplicative gating mechanisms that decouple memory depth from stability constraints [145]. These architectures provide a principled extension of the RNN state-space formulation and are especially well suited for learning evolution operators in multiscale and long-horizon settings. The following section introduces LSTM networks in more detail and discusses their relevance to the operator-learning framework developed in this thesis.

2.5 Long short-term memory networks

The theoretical and practical limitations of classical RNNs, most notably their unstable training dynamics and restricted memory capacity, have motivated the development of gated architectures such as long short-term memory (LSTM) networks [145]. These models introduce multiplicative gating mechanisms that explicitly regulate information flow and gradient propagation, thereby alleviating the spectral constraints that plague standard recurrent networks.

An LSTM augments the classical RNN hidden state with an explicit memory variable, commonly referred to as the *cell state*, which is designed to preserve information over extended time horizons [145, 146]. Let $\boldsymbol{y}(t_k) \in \mathbb{R}^d$ denote the observable system state at discrete time t_k, and let $\mathbf{h}_{t_k} \in \mathbb{R}^p$ and $\mathbf{c}_{t_k} \in \mathbb{R}^p$ denote the hidden and cell states,

respectively. The LSTM dynamics are defined by the coupled system

$$\boldsymbol{f}_k = \sigma\big(W_f \mathbf{h}_{t_k} + \mathbf{U}_f \boldsymbol{y}(t_k) + b_f\big), \tag{2.21}$$
$$\boldsymbol{i}_k = \sigma\big(W_i \mathbf{h}_{t_k} + U_i \boldsymbol{y}(t_k) + b_i\big), \tag{2.22}$$
$$\boldsymbol{o}_k = \sigma\big(\mathbf{W}_y \mathbf{h}_{t_k} + \mathbf{U}_o \boldsymbol{y}(t_k) + \mathbf{b}_y\big), \tag{2.23}$$
$$\tilde{\boldsymbol{c}}_k = \tanh\big(W_c \mathbf{h}_{t_k} + \mathbf{U}_c \boldsymbol{y}(t_k) + b_c\big), \tag{2.24}$$
$$\boldsymbol{c}_{k+1} = \boldsymbol{f}_k \odot \mathbf{c}_{t_k} + \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k, \tag{2.25}$$
$$\boldsymbol{h}_{k+1} = \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_{k+1}), \tag{2.26}$$

where $W_{\{\cdot\}} \in \mathbb{R}^{p\times p}$ and $U_{\{\cdot\}} \in \mathbb{R}^{p\times d}$ are trainable weight matrices, $b_{\{\cdot\}} \in \mathbb{R}^p$ are bias vectors, and $\boldsymbol{f}_k, \boldsymbol{i}_k, \boldsymbol{o}_k, \tilde{\boldsymbol{c}}_k, \mathbf{c}_{t_k}, \mathbf{h}_{t_k} \in \mathbb{R}^p$. The function $\sigma(\cdot)$ denotes the sigmoid activation function, $\tanh(\cdot)$ denotes the hyperbolic tangent activation function, and $\odot$ denotes the Hadamard product.

Equations (2.21)–(2.26) define a nonlinear gated evolution operator on the augmented state space $\mathbb{R}^p \times \mathbb{R}^p \times \mathbb{R}^d$. The coupled LSTM recursions may be written compactly as a single nonlinear state-space system of the form

$$(\mathbf{h}_{t_{k+1}}, \mathbf{c}_{t_{k+1}}) = F_{\mathrm{LSTM}}\big(\boldsymbol{y}(t_k), \mathbf{h}_{t_k}, \mathbf{c}_{t_k}; \boldsymbol{\theta}\big), \tag{2.27}$$

where $\boldsymbol{\theta}$ collects the trainable parameters of the network. The observable prediction is then obtained through an output map

$$\hat{\boldsymbol{y}}(t_{k+1}) = \mathbf{W}_y \mathbf{h}_{t_{k+1}} + \mathbf{b}_y, \tag{2.28}$$

where $\mathbf{W}_y \in \mathbb{R}^{d\times p}$ and $\mathbf{b}_y \in \mathbb{R}^d$. The forget gate regulates memory retention, the input gate controls the incorporation of new information into the cell state, and the output gate determines how much of the internal memory is exposed through the hidden state. This gated structure enables the network to preserve relevant information over long horizons while maintaining stable training dynamics.

Importantly, LSTM networks inherit the universal approximation capabilities of classical recurrent architectures while significantly extending the class of dynamical systems that can be modeled in a stable manner [147]. By decoupling memory preservation from nonlinear state transitions through the introduction of gated additive dynamics, LSTMs are able to represent evolution operators exhibiting long-range temporal dependencies, delay effects, and partially observable latent dynamics.

In this context, this enhanced expressiveness is particularly relevant for multiscale systems, where macroscopic dynamics may depend on slowly evolving latent variables or unresolved microscopic degrees of freedom. From this perspective, the LSTM cell state may be interpreted as a learned latent variable that implicitly encodes subgrid-scale or memory effects not captured by the reduced observable state alone [148]. This interpretation aligns naturally with reduced-order and EF modeling paradigms, in which macroscopic evolution is inferred from data without explicit access to closed-form governing equations.

Despite these many advantages, LSTMs have limitations. The increased architectural complexity introduces a larger parameter space, which may propagate overfitting and reduce interpretability, particularly in regimes with sparse and few data availability. Moreover, while gating mechanisms mitigate gradient issues and improve long-horizon learning, they do not eliminate fundamental challenges associated with chaotic sensitivity, stiff dynamics, or extrapolation beyond the training distribution [149]. As with all data-driven models, the reliability of learned evolution operators ultimately depends on the representativeness of the training data and the validity of the modeling assumptions.

From a numerical analysis viewpoint, the LSTM recursion (2.27) may be interpreted as a data-driven discrete-time integration scheme with adaptive, state-dependent coefficients. The gating variables implicitly regulate the effective time scale of the learned dynamics, enabling the model to emulate both slow and fast processes depending on the input regime. This perspective highlights a conceptual connection between gated recurrent architectures and adaptive time-stepping strategies in classical numerical integration.

In Chapter 4, this general formulation will be specialized to the autoregressive prediction of reduced latent variables, where the observable state corresponds to the reduced representation of the macroscopic dynamics and the operator F_{LSTM} is used to model their temporal evolution.

Finally, while LSTMs provide a powerful nonlinear framework for operator learning, we introduced another powerful linear alternative for operator learning. The following section introduces linear autoregressive frameworks and situates them within the broader landscape of data-driven dynamical system identification.

2.5.1 Linear reduced models via multivariate autoregression

Linear reduced models provide a classical and interpretable framework for approximating the evolution of dynamical systems from data. Within the operator-learning perspective introduced earlier, these models approximate the discrete-time evolution operator by a linear map acting on the reduced coordinate space, possibly with finite memory. Although linearity is a restrictive assumption, such models often serve as robust baselines and, when applied in appropriate reduced or latent coordinates, can capture dominant dynamical features with low computational cost [150, 151, 152].

Linear operator formulation with finite memory

Let $\boldsymbol{y}(t_k) \in \mathbb{R}^d$ denote the vector of latent coordinates at discrete time t_k. A multivariate autoregressive model of order w (MVAR(w)) assumes the linear recursion

$$\boldsymbol{y}(t_k) = \sum_{j=1}^{w} A_j \boldsymbol{y}(t_{k-j}) + \boldsymbol{\varepsilon(t_k)}, \tag{2.29}$$

where $A_j \in \mathbb{R}^{d\times d}$ are constant coefficient matrices and $\boldsymbol{\varepsilon}(t_k) \in \mathbb{R}^d$ represents modeling error, unresolved dynamics, or measurement noise [150, 151]. The error term $\boldsymbol{\varepsilon}(t_k) \in \mathbb{R}^d$

accounts for modeling error, unresolved dynamics, and possible measurement noise. In the classical MVAR formulation, this term is typically assumed to follow a multivariate white noise process. Specifically, the innovations satisfy

$$\mathbb{E}[\boldsymbol{\varepsilon}(t_k)] = \mathbf{0}, \tag{2.30}$$

$$\mathbb{E}[\boldsymbol{\varepsilon}(t_k)\boldsymbol{\varepsilon}(t_k)^\top] = \Sigma, \tag{2.31}$$

$$\mathbb{E}[\boldsymbol{\varepsilon}(t_k)\boldsymbol{\varepsilon}(t_r)^\top] = \mathbf{0}, \qquad k \neq r, \tag{2.32}$$

where $\Sigma \in \mathbb{R}^{d\times d}$ denotes the covariance matrix of the innovations. These assumptions imply that the noise process has zero mean, constant covariance, and no temporal correlation between different time steps.

Under these assumptions, the MVAR model can be interpreted as a linear stochastic dynamical system driven by temporally uncorrelated innovations. The covariance matrix Σ characterizes the magnitude and cross-correlation structure of the residual fluctuations not captured by the linear autoregressive dynamics. This stochastic formulation provides a natural framework for analyzing uncertainty and variability in the latent dynamics [150, 151].

Given a sequence of observations $\{\boldsymbol{y}(t_k)\}_{k=1}^{n_t}$, the coefficient matrices $\{A_j\}_{j=1}^{w}$ can be estimated efficiently using least-squares regression. In particular, stacking the lagged observations into a regression matrix allows the MVAR model to be written in the linear form

$$Y = AX + E, \tag{2.33}$$

where Y collects the current observations, X contains the lagged latent states, and E represents residual errors. This representation admits closed-form estimators and scalable numerical implementations, making MVAR models computationally attractive for large datasets and high-dimensional reduced coordinates.

From a dynamical perspective, the matrices A_j encode the linear temporal interactions among the latent variables. Their eigenstructure determines the dominant growth, decay, and oscillatory modes of the induced linear dynamics. In particular, complex conjugate eigenvalues correspond to oscillatory behavior, while eigenvalues with magnitude smaller than unity indicate asymptotically stable dynamics [150, 151].

Because the coefficients A_j act directly on the reduced coordinates, the MVAR model can be interpreted as a linear surrogate for the unknown evolution operator governing the latent dynamics. When the latent variables provide an effective low-dimensional representation of the system state, such linear autoregressive closures can capture dominant temporal correlations with relatively few parameters.

In this thesis work, MVAR models are employed as linear operator surrogates for the evolution of latent macroscopic coordinates $\boldsymbol{y}(t_k)$. they provide transparent and interpretable representations of temporal dependencies through explicit linear coefficients and finite memory depth. Additionally, they admit optimal estimation via least-squares,

making them computationally efficient for large datasets. Finally, their explicit linear structure enables direct connections to spectral analysis and modal interpretations of the reduced dynamics.

Within the broader operator-learning framework developed in this work, MVAR models serve as structured linear baselines for comparison with nonlinear recurrent architectures such as LSTM networks. By contrasting linear autoregressive models with LSTM-based predictors on the same latent coordinate systems, subsequent chapters assess the extent to which macroscopic dynamics require nonlinear memory representations.

2.6 Lag window size selection via information criteria

An important practical aspect of multivariate autoregressive modeling is the selection of the lag window size w, which determines the temporal memory of the model. Choosing a lag window that is too small may lead to underfitting, failing to capture relevant temporal correlations, whereas overly large lag window orders introduce unnecessary parameters and increase the risk of overfitting. In order to select an appropriate lag order, information-theoretic criteria are commonly employed. In this thesis work we consider two widely used measures: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) [153, 154, 155]. These criteria provide a trade-off between model fit and model complexity by introducing a penalty that increases with the number of model parameters.

For an MVAR model with d latent dimensions and lag window size w, the number of free parameters is $k = d^2 w$. Let $\mathcal{L}(w)$ denote the maximized likelihood of the fitted model. The BIC and AIC scores are then defined as

$$\mathrm{BIC}(w) = -2 \ln \mathcal{L}(w) + k \ln(n_t), \tag{2.34}$$

$$\mathrm{AIC}(w) = -2 \ln \mathcal{L}(w) + 2k, \tag{2.35}$$

where n_t denotes the number of available observations. Both criteria are derived under the assumption that the residual errors follow a Gaussian distribution. Under this assumption, the log-likelihood of the model can be expressed in terms of the determinant of the residual covariance matrix of the fitted MVAR model.

The optimal lag window size is determined by evaluating these criteria over a range of candidate values $w \in \{1, \ldots, w_{\max}\}$ and selecting the lag that minimizes the corresponding score,

$$w^*_{\mathrm{BIC}} = \underset{w \in \{1,\ldots,w_{\max}\}}{\arg\min} \mathrm{BIC}(w), \qquad w^*_{\mathrm{AIC}} = \underset{w \in \{1,\ldots,w_{\max}\}}{\arg\min} \mathrm{AIC}(w). \tag{2.36}$$

The two criteria emphasize different trade-offs. AIC typically favors models with larger lag orders and is often associated with improved short-term predictive performance, whereas BIC imposes a stronger penalty on model complexity and therefore tends to select

more parsimonious models, particularly when the number of observations is large [155]. In practice, both criteria provide complementary guidance for selecting an appropriate memory depth for the autoregressive dynamics.

3 Microscopic Modeling and Data Generation

Abstract This chapter introduces the microscopic modeling framework used throughout this thesis work and describes the generation of data from agent-based simulations of crowd dynamics. Individual pedestrians are modeled as interacting agents whose motion is governed by prescribed equations of motion, interaction mechanisms, and boundary conditions. From the resulting pedestrian trajectories, macroscopic observables, in the form of spatially distributed density fields, are constructed through suitable particle-to-field mappings. As a result, this chapter constitutes the initial step of the modeling framework developed in the thesis, providing the microscopic foundations and macroscopic data representations upon which all subsequent analysis and modeling stages are built.

3.1 Crowds dynamics modeling

Crowd dynamics may be described at different levels of resolution as discussed in Chapter 1, depending on the scale of observation and the quantities of interest. In this thesis work, we distinguish between a *microscopic* level of description, in which individual pedestrians are modeled explicitly, and a *macroscopic* level of description, in which collective behavior is represented through density fields. The relationship between these two viewpoints underlies the multiscale modeling framework developed throughout the thesis and motivates the constructions introduced in this chapter.

At the microscopic level, the crowd is represented as a system of interacting agents, each corresponding to an individual pedestrian. The state of the system at time t is given by the set of agent-level variables, typically including positions and velocities, and the dynamics are specified through interaction mechanisms that govern how each agent responds to neighboring pedestrians, environmental constraints, and external influences. This description is inherently high-dimensional, as the number of degrees of freedom scales with the number of agents in the system. Microscopic models are well suited for capturing heterogeneity, local interactions, and individual decision-making processes, and they provide detailed information about pedestrian trajectories and short-range interactions[15, 119, 156]. In this thesis work, such agent-based models serve as the fundamental dynamical description from which all data are generated.

While microscopic models offer a detailed representation of individual behavior, many questions of interest concern the collective evolution of the crowd at larger spatial and temporal scales. Macroscopic descriptions thus aim to characterize this collective behavior using aggregate quantities that summarize the distribution of pedestrians in space. In this thesis, macroscopic observables are extracted a posteriori from microscopic simulations through suitable particle-to-field mappings, introduced later in this chapter. The coexistence of microscopic and macroscopic viewpoints establishes the foundation for the multiscale modeling pipeline adopted in this thesis work. Microscopic simulations provide detailed trajectory data, while macroscopic representations extracted from these data enable subsequent analysis, reduction, and modeling of collective dynamics.

3.1.1 General microscopic agent description

In this section, we describe the microscopic model used to generate pedestrian trajectory data. The model belongs to the class of interaction-based agent-based descriptions, in which each pedestrian is represented as an individual dynamical system whose motion results from interactions with nearby agents and the surrounding environment. These models are widely used to capture the emergence of collective phenomena in crowd dynamics while maintaining a physically interpretable description at the individual level.

We consider a crowd composed of N pedestrians moving in a two-dimensional spatial domain $\Omega \subset \mathbb{R}^2$. Each pedestrian $i \in \{1, \dots, N\}$ is characterized at time t by its position

$$\mathbf{x}_i(t) = (x_i(t), y_i(t)) \in \Omega \tag{3.1}$$

and velocity

$$\mathbf{v}_i(t) = (v_{ix}(t), v_{iy}(t)) \in \Omega. \tag{3.2}$$

More general crowd models may also incorporate internal or behavioral variables describing cognitive or social traits influencing pedestrian motion. In such cases, the microscopic state of agent/pedestrian i may include a vector of behavioral variables

$$\mathbf{u}_i(t) \in \mathbb{R}^b, \tag{3.3}$$

which can represent quantities such as decision-making strategies, herding tendencies, or panic levels [40, 157, 158, 159, 160]. Within this general framework, the evolution of pedestrian states can be described by interaction-driven ordinary differential equations inspired by pseudo-Newtonian or molecular dynamics formulations [32, 161]. Interactions typically occur locally between pedestrians within a perception or interaction neighborhood. In this thesis work, we adopt a simplified microscopic representation in which the state of each pedestrian is fully described by its position and velocity. The resulting dynamical system therefore takes the form

$$\dot{\mathbf{x}}_i(t) = \mathbf{v}_i(t), \tag{3.4}$$

$$\dot{\mathbf{v}}_i(t) = \mathbf{F}_i(t), \tag{3.5}$$

where $\mathbf{F}_i(t) \in \mathbb{R}^2$ denotes the total force acting on pedestrian i. These forces model the combined effects of motion preferences, interactions with other pedestrians, and interactions with the surrounding environment.

3.1.2 Social Force Model formulation

To specify the interaction mechanisms governing pedestrian motion, we adopt the Social Force Model (SFM), originally introduced by Helbing and Molnár [15]. In this framework, the motion of each pedestrian is driven by a combination of goal-directed behavior, repulsive interactions with other pedestrians, and interactions with static obstacles. The equation of motion of pedestrian i is written as

$$m_i \frac{d\mathbf{v}_i}{dt} = -\frac{1}{\tau_i}(v_{0,i}\mathbf{e}_i - \mathbf{v}_i) + \sum_{j \neq i} \mathbf{F}_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \sum_B \mathbf{F}_{iB}(\mathbf{x}_i), \tag{3.6}$$

where m_i denotes the mass of pedestrian i, $v_{0,i}$ represents the desired walking speed, $\mathbf{e}_i$ is the desired walking direction, and τ_i is a characteristic relaxation time controlling how quickly the pedestrian adapts its velocity toward the desired velocity $v_{0,i}\mathbf{e}_i$. The first term in Eq. (3.6) represents the *goal-directed force*, which models the tendency of each pedestrian to move toward a desired velocity. The second term accounts for interactions with other pedestrians, while the third term represents repulsive interactions with environmental obstacles such as walls or barriers.

3.1.3 Pedestrian–pedestrian interactions

The interaction force between two pedestrians i and j is given by

$$\mathbf{F}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = A_i \exp\left(\frac{r_{ij} - d_{ij}}{B_i}\right)\mathbf{n}_{ij} + k\, g(r_{ij} - d_{ij})\mathbf{n}_{ij} + \kappa\, g(r_{ij} - d_{ij})\Delta v_{ij}^{\text{tang}}\mathbf{t}_{ij}, \quad (3.7)$$

where the first term represents a short-range repulsive force modeling the desire of pedestrians to maintain personal space [15]. The second and third terms represent physical contact forces that become active when pedestrians overlap. The quantities appearing in Eq. (3.7) are defined as follows. The Euclidean distance between pedestrians is

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2, \quad (3.8)$$

while the effective interaction radius is

$$r_{ij} = r_i + r_j. \quad (3.9)$$

The unit vector pointing from pedestrian j to pedestrian i is

$$\mathbf{n}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{d_{ij}}, \quad (3.10)$$

and the tangential unit vector is

$$\mathbf{t}_{ij} = (-n_{ij}^2, n_{ij}^1). \quad (3.11)$$

The function

$$g(x) = \max(0, x) \quad (3.12)$$

is a ramp function ensuring that contact forces are only active when pedestrians overlap, that is when $d_{ij} < r_{ij}$. The first term in Eq. (3.7) models the psychological tendency to maintain interpersonal distance, with strength parameter $A_i > 0$ and decay length $B_i > 0$. The second term models body compression forces arising from physical contact and is controlled by the parameter $k > 0$. The third term represents tangential friction forces during contact and depends on the tangential velocity difference

$$\Delta v_{ij}^{\text{tang}} = \left\|(\mathbf{v}_i - \mathbf{v}_j) - \big((\mathbf{v}_i - \mathbf{v}_j)\cdot \mathbf{n}_{ij}\big)\mathbf{n}_{ij}\right\|_2. \quad (3.13)$$

3.1.4 Pedestrian–obstacle interactions

In addition to interactions with other pedestrians, pedestrians experience repulsive forces from static obstacles such as walls or barriers. The interaction between pedestrian i and an obstacle B is modeled as

$$\mathbf{F}_{iB}(\mathbf{x}_i) = C_i \exp\left(-\frac{d_{iB}}{D_i}\right)\mathbf{n}_{iB} + k\, g(r_i - d_{iB})\mathbf{n}_{iB}, \quad (3.14)$$

where d_{iB} denotes the Euclidean distance between the pedestrian and the closest point of the obstacle, and $\mathbf{n}_{iB}$ is the corresponding outward unit normal vector. The parameter $C_i > 0$ controls the strength of the repulsive interaction, while $D_i > 0$ determines its spatial decay. Several variations of obstacle repulsion models have been proposed in the literature to reproduce specific avoidance behaviors depending on the geometry or type of environment under consideration [130, 162]. Unlike pedestrian–pedestrian interactions, tangential friction terms are typically omitted for static obstacles, as they do not exert dynamic resistance. The parameters appearing in the interaction terms are selected to produce realistic pedestrian trajectories and numerically stable simulations. In the present work, these parameters remain fixed across simulations and serve solely to generate consistent microscopic trajectory data used in the subsequent analysis.

3.1.5 Numerical simulation and data generation

Here, we describe briefly the numerical procedure used to generate microscopic trajectory data from the SFM introduced in Section 3.1.2.

Time discretization and numerical integration

The system of ordinary differential equations governing pedestrian motion is discretized in time using a fixed time step $\delta t > 0$. Let

$$t_k = k\delta t, \qquad k = 0, 1, \ldots, n_t, \tag{3.15}$$

denote the discrete time instants spanning the simulation interval $[0, T]$, where $T = n_t \delta t$. For each pedestrian $i = 1, \ldots, N$, the discrete-time state is given by the pair $(\mathbf{x}_i^k, \mathbf{v}_i^k)$ representing position and velocity at time t_k. The continuous-time dynamics are integrated using a semi-implicit Euler scheme. In particular, velocities are updated using the interaction forces evaluated at the current state,

$$\mathbf{v}_i(t_{k+1}) = \mathbf{v}_i(t_k) + \delta t\, \mathbf{F}_i(\mathbf{x}(t_k), \mathbf{v}(t_k)), \tag{3.16}$$

and positions are then advanced using the updated velocities,

$$\mathbf{x}_i(t_{k+1}) = \mathbf{x}_i(t_k) + \delta t\, \mathbf{v}_i(t_{k+1}). \tag{3.17}$$

We enforce boundary conditions through the interaction forces and the position update. In the longitudinal direction of the corridor, periodic boundary conditions are applied so that pedestrians exiting the domain re-enter from the opposite side, thereby maintaining a sustained flow. Interactions with walls and obstacles are handled through the corresponding force terms in the social force model. The time step δt is chosen sufficiently small to resolve the interaction dynamics and to ensure numerical stability of the integration scheme over the simulation horizon.

Simulation scenarios and initial conditions

We perform simulations in rectangular corridor-like domains $\Omega \subset \mathbb{R}^2$, which contains fixed square obstacles that restrict pedestrian motion. The domain geometry and the placement of obstacles are specified prior to each simulation and remain fixed throughout the simulation. Initial pedestrian positions $\{\mathbf{x}_i(t_0)\}_{i=1}^{N}$ are then prescribed using a variety of spatial distributions, including uniform random distributions, single or multiple Gaussian profiles, and smooth cosine-shaped density patterns. These distributions generate diverse initial crowd configurations while preserving the same underlying microscopic dynamics. All pedestrians are initialized at rest and are assigned a desired direction or target that can be adjusted during the simulation as we will later explain in Chapter 4. The desired motion is encoded through the driving force component of the SFM and numerical integration of the microscopic dynamics produces discrete-time pedestrian trajectories

$$\{\mathbf{x}_i(t_k)\}_{i=1,\dots,N;\ k=0,\dots,n_t}, \tag{3.18}$$

which constitute the microscopic data set used to construct macroscopic density fields in the following section. Additional details regarding domain dimensions, obstacle configurations, parameter values, and the initial distributions are reported in Appendix B.

3.2 From particles to fields: macroscopic observables

The agent-based model (SFM) described in the previous sections provides a collection of microscopic particle trajectories evolving in a prespecified domain for each simulation run. In order to bridge the microscopic description with macroscopic representations suitable for reduced-order modeling, we introduce in this section the macroscopic observable used throughout the remainder of this thesis work: a spatially distributed crowd density field.

The purpose of this section is to define a deterministic particle-to-field mapping that transforms agent-level trajectory data, coming from the SFM simulations, into macroscopic density fields, without introducing any assumed continuum governing equations. These density fields constitute the high-dimensional macroscopic states (ambient space) analyzed in Chapter 4 and modeled dynamically in Chapter 5.

Let $\Omega \subset \mathbb{R}^2$ denote the spatial domain accessible to pedestrians. For a given simulation involving $N \in \mathbb{N}$ agents, the microscopic configuration at time $t \in [0, T]$ is described by the collection of agent positions

$$\mathbf{x}_i(t) \in \Omega, \qquad i = 1, \dots, N. \tag{3.19}$$

A natural macroscopic representation of the particle system is provided by the empirical measure [163].

$$\mu_t := \sum_{i=1}^{N} \delta_{\mathbf{x}_i(t)}, \tag{3.20}$$

where $\delta_{\mathbf{x}}$ denotes the Dirac measure located at $\mathbf{x} \in \Omega$. The empirical measure μ_t encodes the instantaneous spatial distribution of agents. For any measurable set $A \subset \Omega$,

$$\mu_t(A) = \#\{\, i : \mathbf{x}_i(t) \in A \,\}. \tag{3.21}$$

While the empirical measure μ_t provides an exact mathematical description of the particle distribution, it consists of a sum of Dirac masses and is therefore singular with respect to Lebesgue measure. Consequently, it does not admit a regular density representation and is not directly suitable for numerical discretization or geometric analysis. This motivates the introduction of smooth macroscopic density fields obtained by regularizing the empirical measure.

3.2.1 Density as a macroscopic field

We define the macroscopic crowd density as a nonnegative scalar field

$$\rho(\cdot, t) : \Omega \to \mathbb{R}_{\geq 0}, \qquad \rho(\cdot, t) \in L^1(\Omega), \tag{3.22}$$

interpreted as the local concentration of pedestrians (agents per unit area) at time t. Formally, the density field is understood as an absolutely continuous approximation of the empirical measure μ_t in the sense that

$$\mu_t \;\approx\; \rho(\cdot, t)\, d\mathbf{x}, \tag{3.23}$$

where $d\mathbf{x}$ denotes Lebesgue measure on Ω. Equivalently, for sufficiently regular test functions $\varphi : \Omega \to \mathbb{R}$, the density field satisfies the weak approximation property

$$\int_\Omega \varphi(\mathbf{x})\, \rho(\mathbf{x}, t)\, d\mathbf{x} \;\approx\; \sum_{i=1}^{N} \varphi\big(\mathbf{x}_i(t)\big). \tag{3.24}$$

This formulation clarifies the role of $\rho(\cdot, t)$ as a macroscopic observable that reproduces, in an averaged sense, the microscopic spatial statistics of the particle system.

3.2.2 Kernel density estimation

In practice, constructing a smooth macroscopic density field from discrete particle trajectories requires a regularization of the empirical measure. A widely used approach is kernel density estimation (KDE), which approximates the empirical measure by locally averaging the contributions of nearby particles through a smoothing kernel [164, 165]. Given particle positions $\{\mathbf{x}_i(t_k)\}_{i=1}^{N} \subset \Omega$ obtained from the SFM described in Section 3.1.2, the macroscopic density field is defined as follows

$$\rho(\mathbf{x}, t_k) := \sum_{i=1}^{N} K_\xi(\mathbf{x} - \mathbf{x}_i(t_k)). \qquad \mathbf{x} \in \Omega, \tag{3.25}$$

where $K_\xi : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ denotes a smoothing kernel with characteristic length scale $\xi > 0$. Equation (3.25) defines a deterministic particle-to-field mapping

$$\{\mathbf{x}_i(t_k)\}_{i=1}^N \longmapsto \rho(\cdot, t_k), \tag{3.26}$$

which can be interpreted as a regularized approximation of the empirical measure

$$\mu_t = \sum_{i=1}^N \delta_{\mathbf{x}_i(t)}. \tag{3.27}$$

Equivalently, the KDE density corresponds to the convolution

$$\rho(\mathbf{x}, t) = (K_\xi * \mu_t)(\mathbf{x}). \tag{3.28}$$

The kernel K_ξ is assumed to satisfy

- $K_\xi \in L^1(\mathbb{R}^2)$ and $K_\xi \geq 0$,
- normalization $\int_{\mathbb{R}^2} K_\xi(\mathbf{x})\, d\mathbf{x} = 1$,
- sufficient smoothness and radial symmetry.

A common choice is the Gaussian kernel

$$K_\xi(\mathbf{x}) = \frac{1}{2\pi\xi^2} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\xi^2}\right), \tag{3.29}$$

which produces smooth density fields while preserving the total mass of the particle system. The bandwidth parameter ξ controls the spatial resolution of the macroscopic representation. Near boundaries or in the presence of obstacles, kernel averaging may require correction to avoid artifacts caused by kernel support extending outside the admissible region. Such corrections are discussed in Appendix A.

3.2.3 Mass conservation properties

An essential consistency property of the particle-to-field mapping $\mathcal{T}_\rho$ is the preservation of the total number of agents/pedestrians. This property follows directly from the normalization of the kernel function used in the KDE construction. Integrating (3.25) over the spatial domain Ω yields

$$\int_\Omega \rho(\mathbf{x}, t)\, d\mathbf{x} = \int_\Omega \sum_{i=1}^N K_\xi(\mathbf{x} - \mathbf{x}_i(t))\, d\mathbf{x}. \tag{3.30}$$

Since the summation involves a finite number of particles, the order of integration and summation can be exchanged, giving

$$\int_\Omega \rho(\mathbf{x}, t)\, d\mathbf{x} = \sum_{i=1}^N \int_\Omega K_\xi(\mathbf{x} - \mathbf{x}_i(t))\, d\mathbf{x}. \tag{3.31}$$

Under the kernel normalization $\int_{\mathbb{R}^2} K_\xi(\mathbf{x})\, d\mathbf{x} = 1$, each integral evaluates to unity, and therefore

$$\int_\Omega \rho(\mathbf{x}, t)\, d\mathbf{x} = N, \qquad t \in [0, T]. \tag{3.32}$$

Hence, the KDE-based particle-to-field mapping preserves the total mass of the particle system at the continuum level. In numerical implementations, the density field is evaluated on a spatial grid covering Ω. Let δx and δy denote the grid spacings along the coordinate directions. The discrete total mass is then approximated by

$$S(t) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \rho((x_i, y_j), t)\, \delta x\, \delta y. \tag{3.33}$$

This discrete approximation remains close to the number of pedestrians in the system, ensuring that mass conservation is preserved in the numerical representation of the density field. For numerical analysis and reduced-order modeling, the continuous density field $\rho(\cdot, t)$ is evaluated on a fixed spatial grid covering the domain $\Omega = [a, b] \times [c, d]$. Let

$$\mathbf{x} = \{(x_i, y_j)\}_{i=1,\dots,n_x,\ j=1,\dots,n_y} \tag{3.34}$$

denote the grid points associated with a uniform discretization of the domain. The KDE construction (3.25) results then in a discrete density field

$$\rho((x_i, y_j), t), \qquad i = 1, \dots, n_x, \quad j = 1, \dots, n_y, \tag{3.35}$$

which can be arranged into a vector representation

$$\rho(\mathbf{x}, t) = [\rho((x_1, y_1), t), \dots, \rho((x_{n_x}, y_{n_y}), t)]^\top \in \mathbb{R}^{n_c}, \tag{3.36}$$

where $n_c = n_x \times n_y$ denotes the number of spatial degrees of freedom. Evaluating the density field at discrete time instants $t_k = k\Delta t$, obtained by subsampling the microscopic trajectories integrated with time step δt, produces a sequence of macroscopic states $\rho(\mathbf{x}, t_k) \in \mathbb{R}^{n_c}$, for $k = 0, \dots, n_t$, which constitute the observation dataset used in subsequent chapters.

3.3 Discussion

In this chapter, we introduced the microscopic modeling framework used throughout this thesis work and described the construction of macroscopic density fields from agent-based crowd simulations. Starting from pedestrian trajectories generated by the microscopic model (SFM), we defined a deterministic particle-to-field observation operator based on KDE. This procedure results in macroscopic observables that preserve the total number of agents/pedestrians while capturing the collective spatial organization of the crowd. The density fields provide high-dimensional state representations of the system (due to

discretizaion resolution), serving as the macroscopic variables analyzed throughout the remainder of the thesis.

In the next chapter, we investigate the structure of these macroscopic states and examine the extent to which their dynamics can be represented in low-dimensional manifolds suitable for reduced-order modeling.

Part II

Learning Macroscopic Dynamics from data

4 Reduced-Order Representations of Macroscopic Crowd Dynamics

Abstract This chapter develops a reduced order formulation of macroscopic crowd dynamics within an equation-free modeling framework. The macroscopic density fields constructed in Chapter 3 are interpreted as elements of a high-dimensional space describing the collective spatial behavior of the crowd. We analyze the structural properties of this macroscopic space and introduce a physically consistent restriction–lifting framework for constructing reduced representations of density fields. In particular, the restriction operator is defined through linear dimensionality reduction using Proper Orthogonal Decomposition (POD), which maps high-dimensional states (density fields) to low-dimensional latent coordinates. A central feature of this construction is the conservation of mass by design, ensuring that both the reduced states and the reconstructed density fields conserve key physical properties. The resulting reduced state variables provide the foundation for the data-driven identification of macroscopic evolution operators developed in Chapter 5.

4.1 Macroscopic descriptions and closure-based modeling

As mentioned in Chapter 1, macroscopic descriptions of crowd dynamics aim to characterize collective behavior through spatially distributed fields, most commonly pedestrian density and mean velocity. Classical macroscopic models are typically formulated within a continuum framework and expressed as systems of partial differential equations grounded in conservation principles and phenomenological assumptions. A well-known example is the model introduced by Hughes [37], which couples a conservation law for pedestrian density with an Eikonal equation governing route choice through a potential field. Numerous extensions of this framework have been proposed, including nonlocal conservation laws and multi-population models, in which interactions depend on spatially averaged information rather than purely local quantities [27, 32, 38].

From a multiscale perspective, such macroscopic equations are often derived through analytical closure procedures starting from microscopic or mesoscopic descriptions. Kinetic models frequently serve as an intermediate representation, bridging individual-based dynamics and continuum formulations [23, 24, 25, 26]. Within this setting, closure techniques establish relationships between high-dimensional agent-based dynamics or kinetic distributions and a reduced set of macroscopic variables intended to describe observable collective behavior. These approaches provide valuable theoretical insight and, when applicable, yield interpretable macroscopic equations.

At the same time, the derivation and application of closure-based macroscopic models typically require assumptions that are formulated to specific crowd scenarios. These may include idealized interaction mechanisms, homogeneous agent behavior, prescribed functional relationships between velocity and density, or asymptotic limits associated with large populations. As a result, extending a given macroscopic model to new environments, heterogeneous agent populations, or complex geometries often necessitates revisiting the closure assumptions and re-deriving the governing equations. Similar challenges arise in multi-population settings, where ensuring physical consistency, such as per-population mass conservation, while capturing inter-group interactions is nontrivial.

The present thesis work adopts a complementary viewpoint that bypasses the explicit derivation of closed-form macroscopic equations. Instead, macroscopic crowd states are treated directly as high-dimensional objects, density fields evolving in an appropriate function space. This perspective enables an equation-free formulation in which macroscopic density fields are constructed from microscopic simulations, while reduced representations are obtained through mass-preserving mappings.

4.2 Restriction and lifting operators

Having obtained the macroscopic density fields in Chapter 3, we now construct the restriction and lifting operators that connect the high-dimensional macroscopic representation to a low-dimensional latent space. As discussed in Chapter 2, this pair of operators plays a fundamental role in the equation-free framework: the restriction operator provides a reduced representation of the system state, while the lifting operator maps the reduced coordinates back to macroscopic density fields.

In the present context, these operators must be designed to preserve essential physical structure. In particular, since density fields represent macroscopic crowd mass distributions, it is crucial that the restriction–lifting map preserves total mass by construction, rather than relying on a posteriori corrections [166].

The restriction operator via POD

In this context, the restriction operator, say, $\mathcal{R}$ [121], is a map from density distributions $\rho(\mathbf{x}, t)$ to low-dimensional latent space coordinates, say $Y_d(t)$, i.e.,

$$\mathcal{R} : \rho(\mathbf{x}, t) \in \mathbb{R}^{n_c} \mapsto Y_d(t) \in \mathbb{R}^d, \quad d \ll n_c, \tag{4.1}$$

where $n_c = n_x \times n_y$. To construct the restriction operator, we apply POD [138] to a collection of macroscopic density *observations* in time, obtained by microscopic simulations via Eq. (3.25), for different initial conditions; see Appendix B for details. Denoting the densities by $\rho^{(c)}(\mathbf{x}, t_0 + k\Delta t) \equiv \rho^{(c)}(\mathbf{x}, t_k) \in \mathbb{R}^{n_c}$, where Δt is the temporal sampling interval obtained by subsampling the microscopic trajectories integrated with time step δt, t_0 is the initial time, and the superscript (c) denotes the case $c = 1, \ldots, C$ of the different initial conditions considered, the dataset generated by microscopic runs is represented by the matrix $X \in \mathbb{R}^{n_c \times (C \cdot K)}$:

$$X = \left\{ \rho^{(c)}(\mathbf{x}, t_k) \,:\, k = 0, \ldots, K,\ c = 1, \ldots, C \right\}. \tag{4.2}$$

Compactly, we represent the above data set as:

$$X = [x_1, x_2, \ldots, x_{n_t}] \in \mathbb{R}^{n_c \times n_t}, \tag{4.3}$$

with columns, say $x_k = \rho^{(c)}(\mathbf{x}, t_k)/S(t_k) \in \mathbb{R}^{n_c}$, being the normalized density fields along cases and time steps $k = 1, \ldots, n_t = C \cdot K$. Here $S(t_k)$ denotes the total mass at time t_k, defined in Eq. (3.33) in Chapter 3. To complete the restriction operator, we employ POD on X, implemented via Singular Value Decomposition (SVD) on the centered matrix:

$$\bar{X} = XH = U\Sigma V^\top, \quad H = I_{n_t} - \frac{1}{n_t} \mathbf{1}_{n_t} \mathbf{1}_{n_t}^\top, \tag{4.4}$$

where $U \in \mathbb{R}^{n_c \times n_c}$ is the matrix collecting the left-singular vectors, $\Sigma \in \mathbb{R}^{n_c \times n_t}$ is the matrix collecting the singular vectors, $V \in \mathbb{R}^{n_t \times n_t}$ is the matrix collecting the right-singular vectors, I_{n_t} is the n_t-dimensional identity matrix and $\mathbf{1}_{n_t}$ is the n_t-dimensional

unit column vector. The latent space is spanned by the first d left-singular vectors forming U_d. Thus, the projection of X on the latent space provides the low-dimensional embedding:

$$Y_d = U_d^\top \bar{X}, \; Y_d \in \mathbb{R}^{d \times n_t}. \tag{4.5}$$

The lifting operator and mass conservation

For the lifting operator that maps Y_d to the space of density profiles, we demonstrate that the POD reconstruction (i.e., the POD lifting operator) preserves the original total density. This aligns with the general characteristic of POD in maintaining symmetries [167]. We formalize this observation in the following proposition.

Proposition 1. Let us assume a matrix $X \in \mathbb{R}^{n_c \times n_t}$, with columns being normalized density fields. If the density is preserved along time steps, i.e., if:

$$\begin{bmatrix} \sum_{i=1}^{n_c} x_{i,1} & \sum_{i=1}^{n_c} x_{i,2} & \dots & \sum_{i=1}^{n_c} x_{i,n_t} \end{bmatrix} = \mathbf{1}_{n_t}^\top, \tag{4.6}$$

then, the reconstructed density field, computed by the POD as:

$$\tilde{X} = U_d U_d^\top \bar{X} + X(I_{n_t} - H), \tag{4.7}$$

where $U_d = [u_1, u_2, \dots u_d] \in \mathbb{R}^{n_c \times d}$ is the orthonormal basis formed by the first d left-singular vectors computed by the SVD, is also preserved, i.e.,

$$\begin{bmatrix} \sum_{i=1}^{n_c} \tilde{x}_{i,1} & \sum_{i=1}^{n_c} \tilde{x}_{i,2} & \dots & \sum_{i=1}^{n_c} \tilde{x}_{i,n_t} \end{bmatrix} = \mathbf{1}_{n_t}^\top. \tag{4.8}$$

Proof. By construction, since the sum of each column of X is equal to one (that is $\mathbf{1}_{n_c}^\top X = \mathbf{1}_{n_t}^\top$), we have that:

$$\mathbf{1}_{n_c}^\top \bar{X} = \mathbf{1}_{n_c}^\top XH = \mathbf{1}_{n_t}^\top H = \mathbf{1}_{n_t}^\top - \frac{1}{n_t} \mathbf{1}_{n_t}^\top \mathbf{1}_{n_t} \mathbf{1}_{n_t}^\top = \mathbf{1}_{n_t}^\top - \mathbf{1}_{n_t}^\top = \mathbf{0}_{n_t}^\top. \tag{4.9}$$

Therefore, the sum of each column of the centered matrix $\bar{X}$ is equal to zero.

This implies that $\mathbf{1}_{n_c}$ is orthogonal to the column space of $\bar{X}$, and therefore is orthogonal to the left singular vectors of $\bar{X}$, which provide an orthogonal basis for the column space, i.e.,

$$\mathbf{1}_{n_c}^\top U_d = \mathbf{0}_d^\top, \tag{4.10}$$

where $d = 1, 2, \dots, r$ with $r = \text{rank}(X)$. Then, the multiplication of the reconstructed matrix in Eq. (4.7) by $\mathbf{1}_{n_c}^\top$ implies:

$$\mathbf{1}_{n_c}^\top \tilde{X} = \mathbf{1}_{n_c}^\top U_d U_d^\top \bar{X} + \mathbf{1}_{n_c}^\top X(I_{n_t} - H) = \mathbf{0}_{n_t}^\top + \frac{1}{n_t} \mathbf{1}_{n_t}^\top \mathbf{1}_{n_t} \mathbf{1}_{n_t}^\top = \mathbf{1}_{n_t}^\top, \tag{4.11}$$

thus getting:

$$\mathbf{1}_{n_c}^\top \tilde{X} = \begin{bmatrix} \sum_{i=1}^{n_c} \tilde{x}_{i,1} & \sum_{i=1}^{n_c} \tilde{x}_{i,2} & \dots & \sum_{i=1}^{n_c} \tilde{x}_{i,n_t} \end{bmatrix} = \mathbf{1}_{n_t}^\top. \tag{4.12}$$

Hence, the POD reconstruction preserves the total original density. □

In summary, given a density distribution $\rho(\mathbf{x}, t)$ obtained via Eq. (3.25) for a single population, we obtain the normalized density along the n_c grid points of Ω as:

$$x(t) = \frac{1}{S(t)}\left[\rho((x_1, y_1), t), \ldots, \rho((x_{n_x}, y_{n_y}), t)\right]^\top \in \mathbb{R}^{n_c}, \tag{4.13}$$

where $S(t)$ is the total mass at time t, as defined in Eq. (3.33) and define the *restriction* operator $\mathcal{R}$ by the projection on the POD basis of Eq. (4.5) as:

$$\boldsymbol{y}(t) = \mathcal{R}(x(t)) = U_d^\top \left(x(t) - \frac{1}{n_t} X \mathbf{1}_{n_t}\right) \in \mathbb{R}^d, \tag{4.14}$$

where X is the dataset matrix in Eq. (4.3) and U_d is the matrix containing the first d left-singular vectors. On the other hand, the *lifting* operator $\mathcal{L}$ that maps latent coordinates, say $\boldsymbol{y}(t)$, back to the density distributions is defined via the POD reconstruction as:

$$x(t) = \mathcal{L}(\boldsymbol{y}(t)) = U_d \boldsymbol{y}(t) + \frac{1}{n_t} X \mathbf{1}_{n_t} \in \mathbb{R}^{n_c}. \tag{4.15}$$

The construction above provides a linear restriction–lifting pair that is fully specified by the POD basis and the empirical mean of the macroscopic observations. Proposition 1 shows that, under the normalization adopted here, the lifting operator preserves total mass exactly. This property will be exploited in the following to ensure that reduced representations remain physically admissible and that subsequent modeling of macroscopic dynamics in latent space does not introduce spurious mass drift in the reconstructed density fields.

Remark 4.2.1 (Mass conservation versus positivity)**.** Mass conservation and pointwise nonnegativity are distinct properties of macroscopic density reconstructions. While Proposition 1 guarantees that the POD-based lifting operator preserves the total mass of the density field exactly, standard POD truncation does not, in general, enforce pointwise nonnegativity. As a result, small negative values may appear in low-density regions of the reconstructed field.

In the present thesis framework, whenever such negative values arise, they are clipped to zero and the resulting density is renormalized by the total mass of its nonnegative part. Since the POD reconstruction preserves mass by construction, this renormalization compensates only for the negligible mass removed by enforcing positivity, and the correction factor remains close to unity. Consequently, the induced distortion is minimal and localized.

This procedure differs fundamentally from global renormalization strategies applied to non–mass-preserving reconstructions, where normalization compensates for intrinsic mass errors introduced by the reduced representation itself. In such cases, the normalization factor may significantly deviate from unity, uniformly rescaling the entire field and potentially distorting regions that were already accurately reconstructed.

Remark 4.2.2 (Postprocessing and learning are decoupled)**.** The restriction–lifting operators defined in Eqs. (4.14)–(4.15) provide an exact mass-preserving reconstruction in the sense of Proposition 1. When pointwise nonnegativity is enforced in the reconstructed density field through clipping of small negative values (Remark 4.2.1), the resulting renormalization is a purely observational postprocessing step.

Importantly, this postprocessing does not affect the identification of macroscopic dynamics in the proposed pipeline. Learning in Chapter 5 is performed entirely in latent space, using reduced coordinates obtained from the original macroscopic density observations through the restriction operator. Therefore, any nonnegativity enforcement applied after lifting does not modify the latent training data nor the learned evolution operator; it only ensures that reconstructed density fields used for visualization and downstream analysis remain physically admissible.

4.3 Extension to interacting populations via augmented bases

The construction developed in Section 4.2 provides a structure-preserving restriction–lifting pair for macroscopic density fields corresponding to a single pedestrian population. In many realistic crowd scenarios, however, multiple interacting populations coexist within the same spatial domain, as in counterflow or bidirectional movement. In such cases, reduced representations must simultaneously satisfy two competing requirements: they must preserve the total mass of each population separately, and they must capture statistical dependencies between populations that influence the collective dynamics.

A straightforward application of POD independently to each population preserves mass but neglects inter-population correlations. Conversely, joint embeddings that mix population data risk violating per-population conservation. To address this challenge, we extend the POD-based restriction and lifting framework by constructing augmented projection bases that incorporate interaction modes while preserving mass for each population by construction.

Augmented restriction and lifting operators for two populations

We consider the case of two interacting pedestrian groups evolving in the same spatial domain Ω, a typical configuration in dense counterflow settings. As in the single-population case, we collect the macroscopic density fields of each group via Eq. (3.25), and represent them in the observation matrices:

$$X^{(1)} = \left[x_1^{(1)}, \ldots, x_{n_t}^{(1)}\right] \in \mathbb{R}^{n_c \times n_t}, \quad X^{(2)} = \left[x_1^{(2)}, \ldots, x_{n_t}^{(2)}\right] \in \mathbb{R}^{n_c \times n_t}, \tag{4.16}$$

with columns $x_k^{(l)} = \rho_l^{(c)}(\mathbf{x}, t_k)/S_l(t_k) \in \mathbb{R}^{n_c}$ for $k = 1, \ldots, n_t = C \cdot K$, where $l = 1, 2$ indexes the population groups. Applying the single-population POD separately to $X^{(1)}$ and $X^{(2)}$ yields reduced representations that preserve the mass of each group but do not encode statistical coupling between them. To incorporate interaction information,

we construct augmented projection bases that extend the individual POD modes with cross-covariance modes, as formalized in the following proposition.

Proposition 2. Let us assume the observation matrices $X^{(1)}, X^{(2)} \in \mathbb{R}^{n_c \times n_t}$ with columns being normalized density fields of two interacting population groups. Furthermore, let $W \in \mathbb{R}^{n_c \times k}$ and $T \in \mathbb{R}^{n_c \times k}$ be the left and right singular vectors of the cross-covariance matrix

$$C^{(12)} = \frac{1}{n_t} \bar{X}^{(1)} \left(\bar{X}^{(2)} \right)^\top \in \mathbb{R}^{n_c \times n_c}, \tag{4.17}$$

and consider the augmented orthonormal projection bases:

$$\begin{aligned} \tilde{U}^{(1)} &= \left[\tfrac{1}{\sqrt{n_c}} \mathbf{1}_{n_c} \quad U^{(1)}_{d_1} \quad \hat{W} \right] \in \mathbb{R}^{n_c \times (d_1 + k + 1)}, \\ \tilde{U}^{(2)} &= \left[\tfrac{1}{\sqrt{n_c}} \mathbf{1}_{n_c} \quad U^{(2)}_{d_2} \quad \hat{T} \right] \in \mathbb{R}^{n_c \times (d_2 + k + 1)}, \end{aligned} \tag{4.18}$$

where $U^{(1)}_{d_1} \in \mathbb{R}^{n_c \times d_1}$ and $U^{(2)}_{d_2} \in \mathbb{R}^{n_c \times d_2}$ are the first d_1 and d_2 left singular vectors from the SVD of $\bar{X}^{(1)}$ and $\bar{X}^{(2)}$, respectively, and the terms $\hat{W} = W_\perp (W_\perp^\top W_\perp)^{-1/2}$ and $\hat{T} = T_\perp (T_\perp^\top T_\perp)^{-1/2}$ are computed by the orthogonal projections

$$W_\perp = \left(I_{n_c} - \mathbf{1}_{n_c} \mathbf{1}_{n_c}^\top / n_c - U^{(1)}_{d_1} (U^{(1)}_{d_1})^\top \right) W, \tag{4.19}$$

$$T_\perp = \left(I_{n_c} - \mathbf{1}_{n_c} \mathbf{1}_{n_c}^\top / n_c - U^{(2)}_{d_2} (U^{(2)}_{d_2})^\top \right) T. \tag{4.20}$$

If the original densities are mass-preserving along time steps, i.e., if $\mathbf{1}_{n_c}^\top X^{(1)} = \mathbf{1}_{n_t}^\top$ and $\mathbf{1}_{n_c}^\top X^{(2)} = \mathbf{1}_{n_t}^\top$, then the reconstructed densities, computed by projecting $X^{(1)}$ and $X^{(2)}$ onto the augmented bases as:

$$\tilde{X}^{(1)} = \tilde{U}^{(1)} \left(\tilde{U}^{(1)} \right)^\top \bar{X}^{(1)} + X^{(1)} (I_{n_t} - H), \quad \tilde{X}^{(2)} = \tilde{U}^{(2)} \left(\tilde{U}^{(2)} \right)^\top \bar{X}^{(2)} + X^{(2)} (I_{n_t} - H), \tag{4.21}$$

are also mass-preserving, i.e., $\mathbf{1}_{n_c}^\top \tilde{X}^{(1)} = \mathbf{1}_{n_t}^\top$ and $\mathbf{1}_{n_c}^\top \tilde{X}^{(2)} = \mathbf{1}_{n_t}^\top$.

Proof. Assume the original densities are mass-preserving for each group. From the properties of the single-population POD (see Eqs. (4.9) and (4.10)), it follows that:

$$\mathbf{1}_{n_c}^\top \bar{X}^{(l)} = \mathbf{0}_{n_t}^\top, \quad \mathbf{1}_{n_c}^\top U^{(l)}_{d_l} = \mathbf{0}_{d_l}^\top, \quad l = 1, 2. \tag{4.22}$$

Let us consider the SVD of the cross-covariance matrix:

$$C^{(12)} = \frac{1}{n_t} \bar{X}^{(1)} \left(\bar{X}^{(2)} \right)^\top = W \Sigma_{C^{(12)}} T^\top. \tag{4.23}$$

Each column of $C^{(12)}$ is a linear combination of the columns of $\bar{X}^{(1)}$, and each row is a linear combination of the rows of $\bar{X}^{(2)}$. Consequently,

$$\mathcal{R}(W) \subseteq \mathcal{R}(\bar{X}^{(1)}), \quad \mathcal{R}(T) \subseteq \mathcal{R}(\bar{X}^{(2)}). \tag{4.24}$$

To obtain interaction modes orthogonal to both the unity vector $\mathbf{1}_{n_c}$ and the individual POD bases, we project W and T onto the complementary subspace:

$$W_\perp = \left(I_{n_c} - \frac{1}{n_c}\mathbf{1}_{n_c}\mathbf{1}_{n_c}^\top - U_{d_1}^{(1)}(U_{d_1}^{(1)})^\top\right) W, \tag{4.25}$$
$$T_\perp = \left(I_{n_c} - \frac{1}{n_c}\mathbf{1}_{n_c}\mathbf{1}_{n_c}^\top - U_{d_2}^{(2)}(U_{d_2}^{(2)})^\top\right) T.$$

Orthogonalizing these projected vectors yields $\hat{W}$ and $\hat{T}$. By construction,

$$\mathbf{1}_{n_c}^\top \hat{W} = \mathbf{0}_k^\top, \quad \mathbf{1}_{n_c}^\top \hat{T} = \mathbf{0}_k^\top, \tag{4.26}$$

and they are orthogonal to the individual POD bases.

Multiplying the augmented bases by $\mathbf{1}_{n_c}^\top$ from the left and using the above orthogonality relations yields:

$$\mathbf{1}_{n_c}^\top \tilde{U}^{(l)} = \sqrt{n_c}\mathbf{e}_1^\top, \quad l = 1, 2, \tag{4.27}$$

where $\mathbf{e}_1 = [1, 0, \ldots, 0]^\top$. Consequently,

$$\mathbf{1}_{n_c}^\top \tilde{U}^{(l)}(\tilde{U}^{(l)})^\top \bar{X}^{(l)} = \mathbf{1}_{n_c}^\top \bar{X}^{(l)} = \mathbf{0}_{n_t}^\top. \tag{4.28}$$

Finally, using the reconstruction formula in Eq. (4.21) yields:

$$\mathbf{1}_{n_c}^\top \tilde{X}^{(l)} = \mathbf{1}_{n_c}^\top X^{(l)}(I_{n_t} - H) = \mathbf{1}_{n_t}^\top, \quad l = 1, 2, \tag{4.29}$$

which completes the proof. □

The augmented restriction and lifting operators can now be written explicitly. For the restriction operator, we define:

$$\boldsymbol{y}(t) = [\boldsymbol{y}^{(1)}(t), \boldsymbol{y}^{(2)}(t)]^\top \in \mathbb{R}^{d_1+d_2+2k+2}, \quad \boldsymbol{y}^{(l)}(t) = (\tilde{U}^{(l)})^\top \left(x^{(l)}(t) - \frac{1}{n_t}X^{(l)}\mathbf{1}_{n_t}\right), \tag{4.30}$$

and for the lifting operator:

$$x^{(l)}(t) = \tilde{U}^{(l)}\boldsymbol{y}^{(l)}(t) + \frac{1}{n_t}X^{(l)}\mathbf{1}_{n_t}, \quad l = 1, 2. \tag{4.31}$$

In both cases, the restriction and lifting operators guarantee that the reconstructed density fields preserve the total mass of each population, as established in Propositions 1 and 2. While the operators are constructed from the observation matrices $X^{(l)}$, they apply to any seen or unseen macroscopic observation obtained from Eq. (4.13). For conciseness, unless otherwise stated, subsequent developments focus on the single-population notation.

4.4 Discussion

This chapter established a reduced-order representation for macroscopic crowd dynamics in which spatial density fields are treated as high-dimensional system states evolving in time. Starting from microscopic agent-based simulations powered by the SFM, macroscopic observables were constructed and interpreted as elements of a function space whose structure reflects both physical principles and collective behavioral patterns.

A central contribution of this thesis work is the construction of restriction and lifting operators that map between the macroscopic ambient space and a low-dimensional latent representation while preserving mass by design. Unlike many data-driven approaches, where conservation laws are enforced weakly through penalties or regularization in the learning cost function[166], the representations developed here enforce mass conservation structurally through the Restriction and Lifting operators. This distinction is particularly important in crowd dynamics, where density conservation is a fundamental physical requirement and violations may lead to nonphysical predictions or numerical drift in reconstructed macroscopic states.

POD was adopted as the primary restriction operator for several reasons. First, POD provides an optimal linear approximation in the least-squares sense, yielding an efficient low-dimensional representation for a prescribed truncation dimension. Second, the resulting modes admit a clear physical interpretation as dominant spatial density patterns, facilitating qualitative insight into collective behavior. Third, and crucially for the present framework, POD enables exact conservation of total mass under reconstruction, as established in Propositions 1 and 2. These properties make POD particularly suitable for restriction within an equation-free multiscale framework.

At the same time, linear reduced representations have inherent limitations. Although POD captures variance efficiently, it does not account for nonlinear geometric structure in the data. As a result, a larger number of modes may be required to accurately represent dynamics evolving on curved or strongly nonlinear manifolds, especially in heterogeneous or highly interactive crowd scenarios. Moreover, POD truncation does not guarantee pointwise nonnegativity of reconstructed density fields, which may require mild postprocessing when small negative values arise. As discussed in Section 4.2, this adjustment does not affect the learning of latent dynamics but highlights the distinction between mass conservation and positivity as separate structural properties.

The extension to interacting populations further illustrates the trade-offs inherent in reduced representations. Independent POD bases preserve mass but neglect inter-population correlations, while joint representations may violate per-population conservation. The augmented bases introduced here resolve this tension by incorporating interaction modes derived from cross-covariance structures while maintaining exact mass conservation for each population. This construction is essential for realistic multi-population crowd dynamics and constitutes a key methodological contribution of this thesis work.

Importantly, the mass conservation property exploited in this thesis work relies on the fact that total mass is a linear functional of the density field. More specifically, the reduced basis is constructed such that the POD fluctuation modes satisfy the orthogonality condition of Eq. (4.10), which implies that all retained modes belong to the null space of the mass functional. Consequently, the mean density field carries the conserved mass, whereas each retained POD mode has zero contribution to the mass functional. Therefore, the total mass remains invariant under reconstruction.

This construction naturally suggests a pathway for incorporating additional conservation laws. In particular, any conserved quantity that can be expressed as a linear functional of the chosen state variables could, in principle, be enforced through analogous orthogonality constraints on the reduced basis. For example, if the reduced state were augmented to include momentum density variables, conservation of total momentum could be treated in a similar manner by constructing POD modes in the null space of the corresponding momentum functional. In contrast, invariants such as energy are typically nonlinear, often quadratic, functionals of the state. Such quantities are not automatically preserved under linear POD reconstruction, since arbitrary linear combinations of POD modes do not necessarily remain on a constant-energy manifold. Conserving these nonlinear invariants would require additional structure-preserving constraints, constrained projection techniques, or nonlinear reduced representations. Therefore, while the present framework provides a systematic mechanism for enforcing linear conservation laws by construction, extending it to nonlinear invariants constitutes an interesting direction for future research.

Finally, the Restriction and Lifting operators defined in this chapter enable the construction of discrete-time macroscopic evolution operators acting on low-dimensional latent states while guaranteeing physically consistent reconstructions in the ambient space. In this sense, Chapter 4 bridges microscopic simulation data and macroscopic density fields through a reduced-order representation.

5 Learning Reduced-Order Models for the Macroscopic Dynamics

Abstract This chapter is about the construction of data-driven discrete-time evolution surrogate models for macroscopic crowd dynamics in latent spaces. Building on the mass-preserving restriction–lifting framework introduced in Chapter 4, macroscopic density fields are evolved through operators acting on low-dimensional latent coordinates, without postulating explicit macroscopic governing equations. The macroscopic evolution operator is constructed using both linear and nonlinear operator classes introduced in Chapter 2. The learned operators are composed with the lifting operator to reconstruct density fields in ambient space, thereby inheriting exact mass conservation by construction. The framework is validated on representative crowd dynamics scenarios, including unidirectional flow through an obstacle and counterflow of interacting populations. Numerical experiments demonstrate the ability of the learned operators to reproduce macroscopic dynamics and support long-horizon prediction. These results establish a data-driven operator-learning framework for macroscopic crowd dynamics and provide the basis for the state estimation and control tasks developed in subsequent chapters.

5.1 Reduced macroscopic dynamics and global evolution operators

Up to this point, this thesis work has established a systematic framework to bridge the gap between microscopic simulations of crowd dynamics and macroscopic density-based representations. Starting from agent-based simulations, macroscopic observables were constructed in the form of spatial density fields (Chapter 3). Subsequently, we interpreted those fields as elements of a high-dimensional macroscopic space; we constructed both appropriate Restriction and Lifting operators (Chapter 4), enabling their representation in low-dimensional latent coordinates while preserving essential physical properties, such as mass conservation.

The present chapter builds upon this representation framework to address the temporal evolution of macroscopic states. In particular, we adopt the perspective of the *next generation Equation-free (EF) framework* developed similarly in previous work [101, 113, 115]. This framework deals with the curse of dimensionality by learning surrogate machine-learning models in latent spaces emerging from the complex spatio-temporal dynamics of high-dimensional systems.

The approach originates from the classical EF methodology [98], introduced in Chapter 2, which enables high-dimensional simulations to perform system-level numerical analysis tasks on an appropriate latent space. This is achieved via the construction of a *lifting operator* $\mathcal{L}$ and a *restriction operator* $\mathcal{R}$ bridging the gap between states in a high-dimensional space, say $\mathbf{u}(t) \in \mathbb{R}^M$, and states in a latent space, say $\mathbf{y}_d(t) \in \mathbb{R}^d$, with $d \ll M$, as:

$$\mathbf{u}(t) = \mathcal{L}(\mathbf{y}_d(t)) \quad \text{and} \quad \mathbf{y}_d(t) = \mathcal{R}(\mathbf{u}(t)). \tag{5.1}$$

These operators are used to construct a coarse-timestepper, the discrete evolution operator at the latent space, $F_{\delta t} : \mathbb{R}^d \to \mathbb{R}^d$ defined as:

$$\mathbf{y}_d(t + \delta t) = \mathcal{R}\Big(\Phi_{\delta t}\big(\mathcal{L}(\mathbf{u}(t))\big)\Big) := F_{\delta t}\big(\mathbf{y}_d(t)\big), \tag{5.2}$$

where $\Phi_{\delta t} : \mathbb{R}^M \to \mathbb{R}^M$ is the evolution operator at the high-dimensional space. In the traditional EF framework, this coarse evolution is evaluated on demand through short bursts of microscopic simulations. In this thesis work, we instead learn the discrete evolution operator $F_{\delta t}$ using autoregressive reduced-order models (ROMs), thus effectively learning $\Phi_{\delta t}$ as:

$$\mathbf{u}(t + \delta t) = \mathcal{L}\left(\mathbf{y}_d(t + \delta t)\right), \quad \mathbf{y}_d(t + \delta t) = F_{\delta t}\big(\mathbf{y}_d(t), \ldots, \mathbf{y}_d(t - w\delta t)\big), \tag{5.3}$$

where the learned ROMs account for w delayed latent points $\mathbf{y}_d(t - j\delta t)$ for $j = 1, \ldots, w$. The latent points are obtained from previous predictions of $F_{\delta t}$ resulting in a closed-loop/recursive forecasting operator requiring only the initial $j = 1, \ldots, w$ embeddings $\mathcal{R}(\mathbf{u}(j\delta t))$. This formulation enables efficient on-demand reconstruction of the high-dimensional space, as $F_{\delta t}$ evolves in latent space and lifting is performed only when a high-dimensional reconstruction is demanded.

A key distinction between the proposed *next generation EF* framework and the traditional EF methodology is that the latter constructs local numerical maps and computes the quantities needed for system-level analysis through short bursts of microscopic simulations. In contrast, the present approach learns a global dynamical model in Eq. (5.3) via machine learning from long-term, high-fidelity simulations spanning the entire phase space in low-dimensional latent spaces.

The key advantage of this formulation is its ability to circumvent the need for explicit derivation or closure assumptions in macroscopic PDE models of crowd dynamics—a process that inherently introduces modelling biases. Instead, the method directly learns the evolution operator advancing observed macroscopic density distributions forward in time. Embedded within computationally efficient linear or nonlinear autoregressive ROMs, the learned operators encode the complex, often analytically intractable, dynamics of the crowd while enabling efficient prediction of future states solely from observed density snapshots. A schematic overview of the methodology is shown in Fig. 5.1.

5.2 Linear multivariate autoregressive models

In this section, we constrain the global evolution operator in Eq. (5.3) introduced in section 5.1 to a linear class. Specifically, we approximate the reduced macroscopic dynamics by a multivariate autoregressive (MVAR) representation acting on delayed latent coordinates.

As discussed in Chapter 2, the reduced state at discrete time t_k is denoted by $\boldsymbol{y}_c(t_k) \in \mathbb{R}^d$, where the index $c = 1, \ldots, C$ labels distinct realizations corresponding to different initial conditions. To incorporate temporal dependencies, we consider delayed embeddings of width $w \geq 1$, consistent with Takens'/Whitney's embedding theorems [116, 168]. The resulting linear evolution operator is expressed in autoregressive form as

$$\boldsymbol{y}_c(t_k) = \sum_{j=1}^{w} \boldsymbol{A}_j \boldsymbol{y}_c(t_{k-j}) + \boldsymbol{\varepsilon}(t_k), \tag{5.4}$$

where $\boldsymbol{A}_j \in \mathbb{R}^{d \times d}$ are coefficient matrices associated with each lag and $\boldsymbol{\varepsilon}(t_k) \in \mathbb{R}^d$ denotes a residual term. Importantly, we remark that a single ROM is trained in both single- and two-population cases; in the latter, this supports coupled latent dynamics that capture temporal interactions between groups. For compactness in what follows, we denote the dimension of the concatenated two-population latent coordinates in Eq. (4.30) by $d = d_1 + d_2 + 2(m + 1)$, to match the single-population dimensions.

Since the latent coordinates are obtained through mean-centered POD projections, the reduced variables satisfy $\mathbb{E}[\boldsymbol{y}_c(t_k)] = \mathbf{0}$. Consequently, no intercept term is included in Eq. (5.4). The residual term is modeled as a multivariate white noise process satisfying

$$\mathbb{E}[\boldsymbol{\varepsilon}(t_k)] = \mathbf{0}, \tag{5.5}$$

$$\mathbb{E}[\boldsymbol{\varepsilon}(t_k)\boldsymbol{\varepsilon}(t_k)^\top] = \Sigma, \tag{5.6}$$

$$\mathbb{E}[\boldsymbol{\varepsilon}(t_k)\boldsymbol{\varepsilon}(t_r)^\top] = \mathbf{0}, \quad k \neq r, \tag{5.7}$$

for $k, r = 1, \ldots, K$.

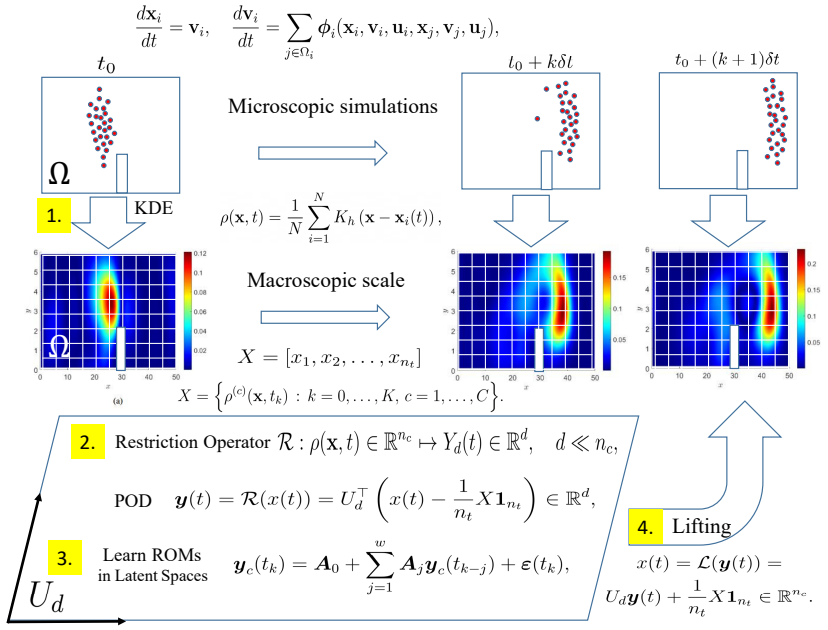


Figure 5.1: Schematic of the proposed methodology. First, microscopic distributions of pedestrian positions are mapped to macroscopic density fields (Step 1) and embedded into a latent space using POD (Step 2). Second, surrogate reduced-order models (MVARs and LSTMs) are trained in this latent space to capture and forecast the complex dynamics (Step 3). Finally, the learned dynamics are lifted back to the high-dimensional density fields via linear projection with the constructed POD basis (Step 4).

5.2.1 Parameter estimation

Given sequential reduced-state data $\{\boldsymbol{y}_c(t_k)\}$, the coefficient matrices $\{\boldsymbol{A}_j\}_{j=1}^{w}$ are determined by minimizing the least-squares objective

$$\underset{\{\boldsymbol{A}_j\}_{j=1}^{w}}{\operatorname{argmin}} \mathcal{L}_{\mathrm{MVAR}}\left(\{\boldsymbol{y}_c(t_k)\}, \{\hat{\boldsymbol{y}}_c(t_k)\}; \{\boldsymbol{A}_j\}_{j=1}^{w}\right), \tag{5.8}$$

where

$$\mathcal{L}_{\mathrm{MVAR}} = \sum_{c=1}^{C} \sum_{k=w+1}^{K} \left\|\boldsymbol{y}_c(t_k) - \hat{\boldsymbol{y}}_c(t_k)\right\|_2^2, \tag{5.9}$$

and the predictions are given by

$$\hat{\boldsymbol{y}}_c(t_k) = \sum_{j=1}^{w} \boldsymbol{A}_j \boldsymbol{y}_c(t_{k-j}). \tag{5.10}$$

The minimization problem in Eq. (5.8) is solved using regularized least-squares methods [169]. In the numerical experiments presented later in this chapter, ridge regularization is employed to mitigate potential overfitting in high-dimensional settings. The lag window size w determines the temporal memory of the operator and must be selected appropriately. In this thesis work, w is chosen using standard information-theoretic criteria; whose details are provided in Chapter 2.

5.2.2 Interpretation and modeling assumptions

The MVAR representation in Eq. (5.4) defines a linear approximation of the global macroscopic evolution operator acting on delayed reduced states. Under this formulation, future latent coordinates are expressed as linear combinations of past coordinates, with the matrices $\boldsymbol{A}_j$ encoding the temporal coupling structure in the reduced space. This linear operator provides a parsimonious and computationally efficient baseline model for macroscopic crowd dynamics. Its identification admits a closed-form least-squares solution, and the resulting coefficient matrices allow for direct interpretation of temporal interactions between reduced modes. The validity of this representation relies on the approximate stationarity of the reduced trajectories. Stationarity can be assessed using standard statistical tests, such as the Augmented Dickey–Fuller (ADF) test [170]. When this assumption is violated, preprocessing procedures such as differencing may be applied prior to identification. Nevertheless, the linear structure may be insufficient to capture strongly nonlinear or non-stationary macroscopic behavior. These limitations motivate the consideration of nonlinear operator classes, introduced in the following section.

5.3 Nonlinear surrogate models

To extend the linear MVAR class of models introduced in Section 5.2, we consider a nonlinear autoregressive representation of the reduced macroscopic dynamics. Specifically, we employ LSTM networks [57, 171], which constitute a class of recurrent neural network architectures designed to model temporal dependencies through gated internal memory states.

Let $\boldsymbol{y}_c(t_k) \in \mathbb{R}^d$ denote the reduced latent state at discrete time t_k, and let w denote the lag window size. The nonlinear evolution operator is defined through an LSTM mapping that processes the past w latent states sequentially. For $\tau = t_{k-w}, \ldots, t_{k-1}$, the internal hidden and memory cell states evolve according to

$$(\mathbf{h}_\tau, \mathbf{c}_\tau) = F_{\text{LSTM}}\big(\boldsymbol{y}_c(\tau), \mathbf{h}_{\tau-1}, \mathbf{c}_{\tau-1}; \boldsymbol{\theta}\big), \tag{5.11}$$

where $\mathbf{h}_{t_k} \in \mathbb{R}^{N_h}$ and $\mathbf{c}_{t_k} \in \mathbb{R}^{N_h}$ denote the hidden and cell states, respectively, and $\boldsymbol{\theta}$ collects the trainable parameters of the recurrent unit. In this thesis work, we employ a vanilla LSTM architecture with four gates, containing parameters $\boldsymbol{\theta} \in \mathbb{R}^{4N_h(d+N_h+1)}$. A detailed description of the LSTM gating mechanism is provided in Chapter 2. The predicted latent state at time t_k is obtained from the last hidden state via a linear output layer,

$$\hat{\boldsymbol{y}}_c(t_k) = \mathbf{W}_y \mathbf{h}_{t_{k-1}} + \mathbf{b}_y, \tag{5.12}$$

where $\mathbf{W}_y \in \mathbb{R}^{d \times N_h}$ and $\mathbf{b}_y \in \mathbb{R}^d$.

5.3.1 Parameter Estimation

Given sequential reduced-state data $\{\boldsymbol{y}_c(t_k)\}$ and a fixed lag window w, the nonlinear operator parameters are identified by minimizing the mean squared error between predicted and true latent states:

$$\underset{\boldsymbol{\theta}, \mathbf{W}_y, \mathbf{b}_y}{\operatorname{argmin}} \, \mathcal{L}_{\text{LSTM}}\left(\{\boldsymbol{y}_c(t_k)\}, \{\hat{\boldsymbol{y}}_c(t_k)\}; \boldsymbol{\theta}, \mathbf{W}_y, \mathbf{b}_y\right), \tag{5.13}$$

with

$$\mathcal{L}_{\text{LSTM}} = \frac{1}{C(K-w)} \sum_{c=1}^{C} \sum_{k=w+1}^{K} \left\| \boldsymbol{y}_c(t_k) - \hat{\boldsymbol{y}}_c(t_k) \right\|_2^2. \tag{5.14}$$

The optimization problem in Eq. (5.13) is solved using gradient-based methods. In the numerical experiments presented in this chapter, we employ the Adam optimizer [172].

5.3.2 Interpretation and modeling assumptions

The LSTM representation defines a nonlinear evolution operator acting on delayed reduced states, with internal memory variables that allow information from previous time steps to influence future predictions. Unlike the linear MVAR operator, this formulation does not impose stationarity assumptions and can accommodate nonlinear temporal interactions in the reduced space. This additional flexibility enables the operator to capture complex and potentially non-stationary macroscopic dynamics. However, the increased expressive capacity comes at the cost of greater computational requirements and reduced interpretability compared to the linear case.

As in Section 5.2, the predicted latent coordinates $\hat{\boldsymbol{y}}_c(t_k)$ are mapped back to the physical density space using the lifting operator defined in Eqs. (4.15) and (4.31) for the single- and two-population cases, respectively. Consequently, reconstructed macroscopic densities inherit the structural mass conservation properties established in Chapter 4, independently of the specific operator class employed.

5.4 Case studies and methodology implementation

In order to demonstrate the capabilities of the pipeline proposed until this part of the thesis. We consider two representative benchmark problems in crowd dynamics:

(i) a unidirectional pedestrian flow moving through a corridor containing an obstacle, and (ii) a counterflow configuration involving two interacting pedestrian populations in the same domain. Both cases are simulated using the microscopic framework introduced in Chapter 3 in subsection 3.1.2 and are used here to validate the reduced macroscopic evolution operators constructed in Sections 5.3 and 5.2.

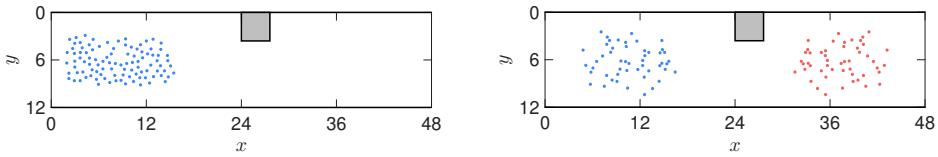
5.4.1 Configurations

Pedestrians move in a corridor of dimensions $48\,\mathrm{m} \times 12\,\mathrm{m}$ (x: length, y: width) in the presence of an obstacle, as illustrated in Fig. 5.2.

The square obstacle of area $12.96\,\mathrm{m}^2$ is centered at $(25.8\,\mathrm{m}, 1.8\,\mathrm{m})$. The computational domain is $\Omega\backslash\Gamma$, where $\Omega = [0, 48] \times [0, 12]$ and $\Gamma = [24, 27.6] \times [0, 3.6]$. Pedestrian motion is modeled using the SFM [15, 16], which falls within the microscopic class introduced in Chapter 3 in subsection 3.1.2:

$$m_i \frac{d\boldsymbol{x}_i}{dt} = -\frac{1}{\tau_i}(v_{0i}\boldsymbol{e}_i - \boldsymbol{v}_i) + \sum_{j \neq i} \mathbf{F}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{\text{obstacles}} \mathbf{F}_{iB}(\boldsymbol{x}_i). \tag{5.15}$$

Random forces are omitted to focus on deterministic collective behavior. Full SFM details are provided in Chapter 3 in subsection 3.1.2. In the unidirectional flow configuration (see Fig. 5.2a), we simulate a crowd of $N = 100$ pedestrians moving through a



(a) Unidirectional flow

(b) Counterflow

Figure 5.2: Crowd dynamics configurations of SFM simulating pedestrians moving in a corridor past an obstacle. In the unidirectional configuration, pedestrians move from left to right. In the counterflow configuration, two pedestrian populations move in opposite directions; group 1 (blue) moves from left to right, while group 2 (red) moves in the opposite direction.

corridor, with initial spatial distributions ranging from homogeneous dispersions to clustered configurations (for more details, see Appendix B). Pedestrians are directed to move from the left to the right of the corridor. Since the SFM governs pedestrians' motion through target-directed forces without intrinsic path planning, we employ a sequential two-target scheme to induce realistic obstacle-avoidance behavior. Pedestrians are initially guided towards an intermediate zone below the obstacle at coordinates $x = 25\,\text{m}$ and $y \in [6, 7.8]\,\text{m}$, to navigate around the obstacle. Upon reaching this zone, their final destination is updated to $x = 48\,\text{m}$ with y-coordinates remaining at the values each agent had upon reaching the intermediate zone. To maintain a constant population density, we implement periodic boundary conditions by re-spawning agents at the left boundary at $x = 0\,\text{m}$, when agents reach the right boundary at $x = 48\,\text{m}$.

In the counterflow configuration (see Fig. 5.2b), we simulate a crowd of $N = 100$ pedestrians divided equally into two groups with opposing flow directions: group 1 moves from left to right, and group 2 from right to left. We use the SFM in Eq. (5.15) with the same parameters as in the unidirectional case, and initial spatial distributions that are symmetric for each group along $x = 24\,\text{m}$; group 1 follows the same initial distribution as in the unidirectional case. Employing a two-target scheme to induce realistic obstacle-avoidance behavior, we prescribe distinct preferred zones for each group. Group 1 is initially guided to the zone $x = 25\,\text{m}$ and $y \in [5.4, 7.8]\,\text{m}$, while group 2 to $x = 25\,\text{m}$ and $y \in [9.6, 11.4]\,\text{m}$. Upon reaching these zones, their final destination is updated to $x = 48\,\text{m}$ for group 1 and $x = 0\,\text{m}$ for group 2, while the y-coordinates remain fixed at the values each agent had upon reaching the intermediate zone. This separation promotes smooth obstacle avoidance and organized counterflow without introducing explicit route planning. As in the unidirectional case, periodic boundary conditions are applied at both ends of the corridor by re-spawning agents at the left/right boundary upon exiting at the right/left for groups 1/2, respectively.

In both configurations pedestrians are prohibited from passing the corridor's walls or entering the obstacle area by introducing strong wall forces in the SFM microscopic simulator. Such a configuration ensures a controlled pedestrian flow through the simulated

environment and ensures a realistic obstacle-avoidance behavior.

Remark 5.4.1. Here, we employed the SFM purely as a convenient baseline to demonstrate and validate the proposed methodology. Naturally, the same workflow is not tied to this specific choice: it can be coupled just as well with more sophisticated agent-based approaches, e.g., models that include explicit route planning, richer interaction rules, heterogeneous behaviors, or learned decision-making, depending on the fidelity required by the application.

5.4.2 Data generation of microscopic distributions

We generate synthetic data of pedestrian trajectories by numerically integrating the SFM with a first-order explicit Euler scheme using a small fixed time step of $\delta t_{micro} = 0.025$ s to ensure stability and accuracy. Each simulation has a time span of 250 s. To reduce computational storage costs while retaining essential temporal features, we subsample the resulting pedestrian trajectories at intervals of $\delta t = 0.25$ s. As already discussed, we generate pedestrian trajectories from various initial spatial distributions modelling diverse crowd scenarios. In particular, we consider $C = 20$ different initializations with zero velocities and positions distributed via: (i) uniform distributions to provide a homogeneous baseline scenario, (ii) Gaussian clusters to model localized groups of pedestrians, (iii) "double bell-shaped" Gaussian distributions to capture the formation of two distinct crowd groups and (iv) cosinusoidal distributions to generate periodic macroscopic behavior. Details on the form and parameters of the above initializations are provided in Appendix B, distinguishing between those used for training and testing the proposed framework in Tables B.1 and B.2, respectively. For the unidirectional flow study case: the resulting datasets are comprised of 20 cases with different initializations, each containing a matrix $\mathcal{X}^{(c)} = \{x_i(t_k), i = 1, \dots, N, k = 1, \dots, K\} \in \mathbb{R}^{200\times1000}$, where $2N = 200$ corresponds to the pedestrians' positions and $K = 1000$ time steps correspond to the samples recorded from each trajectory. Thus, the total datasets over all cases are $\mathcal{X}_{tr}, \mathcal{X}_{ts} \in \mathbb{R}^{200\times(20\cdot1000)}$ used for training and testing, respectively.

In counterflow study case, we generate data from the same set of $C = 20$ initializations; see Appendix B. Here, the initialization for group 1 is identical to that of the unidirectional case (with half the agents), while group 2 is initialized symmetrically to group 1 along $x = 24$ m, preserving the y-positions. Each initialization case now produces two separate data matrices $\mathcal{X}^{(1,c)}$ for group 1 and $\mathcal{X}^{(2,c)}$ for group 2, both of dimension $\mathbb{R}^{100\times1000}$. Consequently, the complete training and testing datasets for the counterflow are $\{\mathcal{X}_{tr}^{(1)}, \mathcal{X}_{tr}^{(2)}\}$ and $\{\mathcal{X}_{ts}^{(1)}, \mathcal{X}_{ts}^{(2)}\}$, respectively.

5.4.3 Extraction of continuous density fields

For the first step of the proposed framework, we derive macroscopic density fields from pedestrian positions via KDE, as described in Chapter 3 in subsection 3.2.2. To achieve this, we discretize the corridor computational domain Ω into $n_x \times n_y = 80 \times 20$ control volumes with uniform spacing $\delta x = \delta y = 0.6$ m; this resolution reflects a typical personal space for each pedestrian, allowing for capturing emergent collective behavior, while minimizing artificial grid effects. For the KDE implementation on the cell centroids, we choose Gaussian kernels in Eq. (3.25) tuned with bandwidths $\mathbf{H} = diag(h_x, h_y) = (3, 2)$ for suppressing noise and preserving features of the microscopic data, while ensuring that local structures remain clear without introducing spurious artifacts. To account for the physical obstacle in $\Gamma \subset \Omega$, we apply a binary mask to the KDE-estimated density field, enforcing zero density in Γ. This is necessary because KDE can produce non-zero density estimates near obstacle boundaries when pedestrians are in close proximity. To account for the periodic boundary conditions, we further perform domain augmentation for the KDE support. All the details regarding KDE implementation are provided in Appendix A.

Using the above procedure, in the unidirectional flow case we derive the density fields $\rho^{(c)}(x_i, y_j, t_k)$ ($c = 1, \ldots, C$, $i = 1, \ldots, n_x$, $j = 1, \ldots, n_y$ and $k = 1, \ldots, K$) from the pedestrian positions included in the training $\mathcal{X}_{tr}$ and testing $\mathcal{X}_{ts}$ data sets. For conforming to the assumptions of Proposition 1 in Chapter 4, we normalize each density field following Eq. (4.13), thus constructing the normalized density matrices $X_{tr}, X_{ts} \in \mathbb{R}^{n_c \times n_t}$ in the form of Eq. (4.3), where $n_c = n_x \times n_y = 1600$ and $n_t = C \cdot K = 20 \cdot 1000$. In the counterflow case, we followed the same procedure to extract and normalize the density fields of each group separately, as it is required by Proposition 2 in Chapter 4, yielding the matrices $X_{tr}^{(1)}, X_{ts}^{(1)} \in \mathbb{R}^{n_c \times n_t}$ for group 1 and $X_{tr}^{(2)}, X_{ts}^{(2)} \in \mathbb{R}^{n_c \times n_t}$ for group 2.

5.4.4 POD for the restriction and lifting operators

We construct the restriction and lifting operators by employing POD on the training data, determining the latent dimension d and the POD basis U_d via the economy-size SVD. In all cases, we determine d by the best low-rank approximation retaining 99% of the energy; thus, computing the minimum number satisfying the criterion:

$$E_d = \frac{\sum_{i=1}^{d} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \geq 0.99 \tag{5.16}$$

where $r = \min(n_c, n_t) = 1600$ is the rank of the data matrix, and $\sigma_i > 0$ for $i = 1, \ldots, r$ are the singular values in descending order. The basis U_d is then formed by the first d left singular vectors.

For the unidirectional flow study case, we apply this procedure to the training dataset X_{tr} and the resulting basis U_d is then used to construct the restriction and lifting operators in Eqs. (4.14) and (4.15), respectively.

On the other hand, for the counterflow study case: the above procedure is done per group, to compute the separate POD bases $U_{d_1}^{(1)}$ and $U_{d_2}^{(2)}$ from the group-specific training datasets $X_{tr}^{(1)}$ and $X_{tr}^{(2)}$, respectively. Following Proposition 2 in Chapter 4, we then compute the SVD of the cross-covariance matrix $C^{(12)} = \frac{1}{n_t}\bar{X}_{tr}^{(1)}\left(\bar{X}_{tr}^{(2)}\right)^{\top}$ of the training sets, which is required to form the augmented bases $\tilde{U}_{d_l}^{(l)}$ (for $l = 1, 2$) defined in Eq. (2). These augmented bases are subsequently used to construct the group-specific restriction and lifting operators in Eqs. (4.30) and (4.31).

For both configurations, to assess the accuracy of restriction and lifting operators, we evaluate the POD reconstruction error on the training and testing data sets by the relative L_2 error:

$$e_k^{2,rec} = \frac{\|x(t_k) - \mathcal{L}(\mathcal{R}(x(t_k)))\|_2}{\|x(t_k)\|_2}, \tag{5.17}$$

where $x(t_k) \in X_{tr}, X_{ts}$ denotes the normalized density in Eq. (4.13) at time t_k for $k = 1, \ldots, K$. For the counterflow, this error is computed separately for each group using the corresponding operators and data sets $X_{tr}^{(l)}, X_{ts}^{(l)}$ for $l = 1, 2$.

5.4.5 ROMs for latent space dynamics and their training

Having obtained the latent representations $\boldsymbol{y}(t) \in \mathbb{R}^d$ from the restriction operator, the third step of the proposed approach is to construct ROMs for learning the dynamics in the latent space. For this purpose, we learn linear MVAR and non-linear LSTM models, as described in sections 5.2-5.3, for predicting the latent variable $\hat{\boldsymbol{y}}_c(t_k)$ of the c case at timestep t_k given the past w lags $\{\boldsymbol{y}_c(t_{k-w}), \ldots, \boldsymbol{y}_c(t_{k-1})\}$. Note that in the counterflow, a single ROM is trained on the concatenated latent vectors from both groups $\boldsymbol{y}_c(t_k) = [\boldsymbol{y}_c^{(1)}(t_k), \boldsymbol{y}_c^{(2)}(t_k)]^{\top}$, as detailed in in sections 5.2-5.3. To conform to sequence-to-one learning, the features consist of the dataset $Y_f = \{\boldsymbol{y}_c(t_{k-w}), \ldots, \boldsymbol{y}_c(t_{k-1})\} \in \mathbb{R}^{d\times w\times C\cdot(K-w)}$ and the targets consist of the data set $Y_t = \{\boldsymbol{y}_c(t_k)\} \in \mathbb{R}^{d\times C\cdot(K-w)}$, where each $\boldsymbol{y}_c(t_k)$ is computed via the restriction operator in Eq. (4.14) on the training data X_{tr} (for the unidirectional case) and via Eq. (4.30) on $X_{tr}^{(1)}, X_{tr}^{(2)}$ (for the counterflow case).

As a preliminary step, we first tune the lag window w, within a bound of $w_{\max} = 20$, using two information-theoretic criteria: the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC) [155, 153, 154]; details are provided in Chapter 2. Both criteria are likelihood-based, with AIC tending to select larger windows than BIC. We note that BIC and AIC are well-suited for MVARs since they assume stationarity of the data and white noise residuals. For comparison, we adopt the same w for learning LSTMs, as the one tuned for MVARs by these criteria.

For the MVAR model training in both configurations, we first verify that all input time series are stationary using the augmented Dickey–Fuller (ADF) test [173] with a threshold of significance level $p < 0.01$. Then, we compute the coefficients $\boldsymbol{A}_j$ in Eq. (5.4) using

ordinary least squares to solve the optimization problem in Eq. (5.8). For the counterflow case, where the related data-matrix was ill-conditioned, we employed ridge regularization with $\lambda = 10^{-6}$.

For the LSTM model in both configurations, we consider a single LSTM layer with 16 units and a hyperbolic tangent activation function in the hidden layers; the output layer is linear, as denoted in Eq. (5.12). To train the LSTM model, we solve the optimization problem in Eq. (5.13) using the Adam optimizer [172] with automatic differentiation for the gradients of the loss function, enabled through MATLAB's Deep Learning Toolbox [174]. The model's parameters are initialized with a Glorot uniform initialization [175], the initial learning rate is set to $l_r = 10^{-3}$, and the mini-batch size is set to 32; the latter hyperparameters were tuned on a trial-and-error basis, providing a good compromise between computational efficiency and generalization without exceeding memory constraints. The maximum number of training epochs was set to 40 for the unidirectional flow case and to 70 for the counterflow case, based on the saturation of the training loss, which yielded no additional improvement beyond these epochs.

To assess the convergence of both MVARs and LSTMs training, we report the values of the loss functions in Eqs. (5.9) and (5.14), respectively, at the end of training. To further quantify the uncertainty associated with the stochastic training of the LSTM models, we repeated their training procedure 50 times, each with different randomly initialized learnable parameters. We report the mean error of the final losses and the 10–90% error percentiles across the 50 runs. Since the MVARs loss function is based on L_2 error, we further report the MSE at the latent space:

$$MSE_{\text{MVAR}}\left(\{\boldsymbol{y}_c(t_k)\},\{\hat{\boldsymbol{y}}_c(t_k)\};\{\boldsymbol{A}_j\}_{j=1}^{w}\right)=\frac{1}{C(K-w)}\sum_{c=1}^{C}\sum_{t_k=w+1}^{K}\left\|\boldsymbol{y}_c(t_k)-\hat{\boldsymbol{y}}_c(t_k)\right\|_2^2, \tag{5.18}$$

where $\hat{\boldsymbol{y}}_c(t_k)$ is the prediction in Eq. (5.10), to enable direct comparison with the MSE-based loss function of LSTMs. Additionally, we report the computational times required for training both ROMs (for LSTMs, we report the mean error and 10–90% error percentiles statistics across the 50 runs); we expect MVARs to be much faster than LSTMs due to the lower number of parameters and the method employed for solving the related optimization problem.

5.4.6 Accuracy assessment of the proposed framework

6Having constructed/trained each step of the proposed framework (construction of the restriction/lifting operators and training of the ROMs), we finally assess the forecasting performance of the whole framework on the $C = 20$ different initializations of the testing set $\mathcal{X}_{ts}$; see Appendix B for details. In particular, we first obtain the "ground truth" normalized densities for each initialization, say $x^{(c)}(t_0 + k\ \delta t) \equiv x^{(c)}(t_k) \in X_{ts}$ for all time steps $k = 1, \ldots, K$, using Eq. (4.13). While ROMs are trained in an open-loop setting using ground truth embeddings, we evaluate the proposed framework in a closed-loop setting (i.e., recursively feeding subsequent predictions to the autoregressive ROMs),

as this reflects the intended use in practice. In particular, following the restrict-evolve with ROM-lift framework in Eq. (5.3), we compute the closed-loop prediction:

$$\hat{x}^{(c)}(t_k) = \mathcal{L}\left(\hat{\mathbf{y}}^{(c)}(t_k)\right), \qquad \hat{\mathbf{y}}^{(c)}(t_k) = \Phi_{\text{ROM}}\left(\hat{\mathbf{y}}^{(c)}(t_{k-1}), \ldots, \hat{\mathbf{y}}^{(c)}(t_{k-w}); \mathbf{p}\right), \tag{5.19}$$

where $\hat{\mathbf{y}}^{(c)}(t_{k-j})$ are latent points predicted recursively by Φ_{ROM} in the previous $j = 1, \ldots, w$ steps. Closed-loop forecasting requires the initial $j = 1, \ldots, w$ embeddings of the observed densities $\mathcal{R}(x^{(c)}(t_j))$. Here, $\mathcal{R}$ and $\mathcal{L}$ are the restriction and lifting operators in Eqs. (4.14) and (4.15) (for the unidirectional flow) or in Eqs. (4.30) and (4.31) (for the counterflow) and Φ_{ROM} is the latent dynamics ROM with parameters $\mathbf{p}$ that can be either the trained MVAR or LSTM model; for LSTM, we select the best-performing model among the 50 training runs. For completeness, we provide the open-loop setting and corresponding one-step prediction results in Appendix C and D for the unidirectional flow and counterflow cases, respectively. In open-loop, the latent points are obtained from ground truth embeddings $\mathbf{y}^{(c)}(t_{k-j}) = \mathcal{R}(x^{(c)}(t_{k-j}))$ rather than previous predictions $\hat{\mathbf{y}}^{(c)}(t_{k-j})$ in Eq. (5.19).

For evaluation purposes, the prediction is obtained at every time step k. However, in practice, the lifting is applied on-demand only when a high-dimensional reconstruction is required. This renders the scheme particularly efficient, as the evolution is governed by the ROM in the latent space. To quantify the end-to-end prediction accuracy, we compute the relative, reconstructed in the ambient-space, L_1, L_2 and L_∞ errors between the "ground truth" densities $x^{(c)}(t_k)$ and the predicted ones $\hat{x}^{(c)}(t_k)$ per case c in the testing set and time step k as:

$$\begin{aligned} e_k^{1,(c)} &= \frac{\|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)\|_1}{\|x^{(c)}(t_k)\|_1}, \\ e_k^{2,(c)} &= \frac{\|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)\|_2}{\|x^{(c)}(t_k)\|_2}, \\ e_k^{\infty,(c)} &= \frac{\|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)\|_\infty}{\|x^{(c)}(t_k)\|_\infty}. \end{aligned} \tag{5.20}$$

For a summary of statistics, we report the mean and the 10–90% percentiles of the above errors *over all cases and time steps*. To track down the prediction accuracy in time, we further depict the mean and 10–90% percentiles of the error evolution in time *over all cases*.For the counterflow, these errors are computed separately for each group $l = 1, 2$. All numerical computations were executed on a PC equipped with an 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 GHz CPU and 16 GB of system memory. The source code is publicly available at `github.com/patsatzisdim/NGEF_CD`.

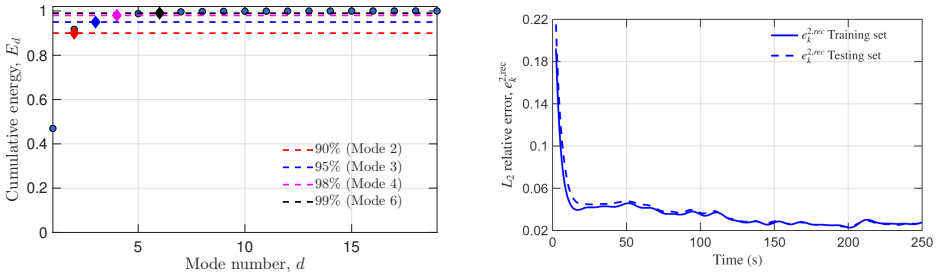
5.5 Numerical results

We present results of the proposed framework for predicting pedestrian flow in a rectangular corridor with an obstacle under the unidirectional and counterflow configurations described in Section 5.4.1. The section concludes with a comparative discussion of the MVAR and LSTM models.

5.5.1 Unidirectional flow case study

We first consider the unidirectional flow of pedestrians moving from the left to the right of the corridor as shown in Fig. 5.2a. As already discussed in Section 5.4.2, we first collected pedestrian trajectory data using the SFM microscopic agent-based simulator under a wide range of initial conditions. The resulting training dataset $\mathcal{X}_{tr}$ is used to train our framework, while the testing dataset $\mathcal{X}_{ts}$ is used to assess its efficiency and prediction accuracy.

In the first step, we extracted the normalized density fields data X_{tr}, X_{ts} from the discrete pedestrian positions in $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ using KDE, as described in Section 5.4.3. Next, we employed POD on $\mathcal{X}_{tr}$ to discover the latent space dimension; see Section 5.4.4. Following the criterion in Eq. (5.16), the minimum latent dimension required for retaining more than 99% of the energy in the training dataset is $d = 6$ modes, as shown in Fig. 5.3(a). Using the discovered POD basis U_d, we next constructed the restriction and



(a) POD cumulative energy

(b) POD reconstruction error

Figure 5.3: Accuracy of the restriction and lifting operators in Eqs. (4.14) and (4.15) for the unidirectional flow case. Panel (a) shows the minimum number of POD modes (colored diamonds) required for retaining the desired percentage of energy in the training data set. Panel (b) depicts the average (over the $C = 20$ cases with different initial conditions) reconstruction error $e_k^{2,rec}$ in Eq. (5.17) with $d = 6$ POD modes, in time for the training and testing datasets.

lifting operators in Eqs. (4.14) and (4.15). To assess their accuracy, we computed the relative L_2 reconstruction error $e_k^{2,rec}$ in Eq. (5.17) across the $k = 1, \ldots, K$ time steps of each trajectory. Fig. 5.3(b) displays the average relative error *over all the* $C = 20$ cases with different initial conditions included in either the training or testing datasets.

While initially the reconstruction error is ∼21%, it decreases to 4% at the first 20 s and further reduces to ∼3% after 50 s for both seen (training set) and unseen (testing set) data, indicating high accuracy of the restriction and lifting operators with a significantly reduced latent space dimension $d = 6$. We note that this level of reconstruction error serves as an *infimum approximation error baseline*, i.e., the best possible error for the proposed framework at the ambient–density profiles–space. ROM forecasts in the long term are therefore expected to surpass this baseline error due to the additional error contributions from model-form error and training/generalization errors.

We next train MVAR and LSTM models to learn ROMs for the latent dynamics (see Sections 5.2 and 5.3), where the latent variables $\boldsymbol{y}_c(t_k) \in \mathbb{R}^6$ of the c case at timestep t_k are obtained from the density profiles of the training set via the restriction operator in Eq. (4.14). To determine the lag window, we employed the BIC and AIC criteria described in 2.6; BIC indicated a lag window of $w = 4$, whereas AIC favored a wider (as expected) $w = 9$. Both windows were used for training both MVAR and LSTM models, denoted from now on as MVAR(4), MVAR(9), LSTM(4), and LSTM(9) models. The training data sets were divided into features $Y_f = \{\boldsymbol{y}_c(t_{k-w}), \ldots, \boldsymbol{y}_c(t_{k-1})\}$ and targets $Y_t = \{\boldsymbol{y}_c(t_k)\}$.

For the MVARs training, we first verified that the latent dynamics datasets Y_f, Y_t constitute stationary time series; the ADF test confirmed stationarity with a significance level of $p < 0.001$ for all d variables. We next trained all MVAR and LSTM models using algorithms, hyperparameters and stopping criteria discussed in detail in Section 5.4.5. The training results are shown in Table 5.1 for the four ROMs, including the loss function values attained at the end of training and the computational time (in seconds) required. For the LSTMs, as each model was trained over 50 runs with different parameter initializations, we report in Table 5.1 the mean and the 10–90% error percentiles across the 50 runs. Since the loss function of MVARs is L_2-based, we additionally compute the

Table 5.1: Training results of MVAR(4), MVAR(9), LSTM(4) and LSTM(9) models at the latent space for the unidirectional flow case. Loss functions $\mathcal{L}_{\text{MVAR}}(\cdot)$ in Eq. (5.9) for MVARs and $\mathcal{L}_{\text{LSTM}}(\cdot)$ in Eq. (5.14) for LSTMs are reported at the end of training, along with computational times (in seconds) required for training. For the LSTMs, the mean and 10–90% error percentiles (in parentheses) are reported across the 50 training processes with different parameter initializations. Additionally, the MSE loss function of MVARs in Eq. (5.18) is reported for comparison to the MSE-based loss $\mathcal{L}_{\text{LSTM}}(\cdot)$.

Model	$\mathcal{L}_{\text{MVAR}}(\cdot)$	$MSE_{\text{MVAR}}(\cdot)$	$\mathcal{L}_{\text{LSTM}}(\cdot)$	Comput. Time (s)
MVAR(4)	4.65×10^{-8}	3.89×10^{-13}	-	0.0013
MVAR(9)	2.04×10^{-8}	1.72×10^{-13}	-	0.0017
LSTM(4)	-	-	$7.12\ (6.27, 8.16) \times 10^{-9}$	62 (55, 66)
LSTM(9)	-	-	$1.51\ (1.30, 1.81) \times 10^{-8}$	86 (77, 89)

MSE in Eq. (5.18) to enable direct comparison with the MSE-based LSTM loss. MVARs

converge to much smaller loss function values in comparison to LSTMs. As expected, LSTMs require significantly higher computational time for training in comparison to MVARs. Table 5.1 further shows that the training of both types of ROMs with increased lag window requires more computational time.

5.5.2 Recursive/Closed-loop Predictions

Having constructed and trained all components of our framework, we now evaluate its end-to-end performance in the ambient–density profile–space. While the ROMs are trained in an open-loop setting, we assess them in the practical closed-loop setting (see Eq. (5.19)), where only the first w observations are supplied and all subsequent predictions are generated autoregressively. For LSTM models, we consider the best-performing model from the 50 training runs. We quantify the prediction accuracy via the relative L_1, L_2, and L_∞ errors, $e_k^{1,(c)}$, $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ in Eq. (5.4.6), computed between the "ground-truth" density profiles $x^{(c)}(t_k)$ and the predicted/reconstructed ones $\hat{x}^{(c)}(t_k)$ for the C, unseen in training, cases (of different initial conditions) included in the testing set X_{ts} across time steps k.

Table 5.2 provides a summary of the closed-loop reconstructed errors, including the mean and the 10–90% error percentiles, aggregated *over all cases and time steps*. As shown in

Table 5.2: Closed-loop/recursive prediction relative errors in the ambient–density profile–space, for the testing set using the trained MVAR and LSTM (best out of 50 training runs) models for the unidirectional flow case. The relative reconstructed L_1, L_2 and L_∞ errors $e_k^{1,(c)}$, $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ in Eq. (5.4.6) are reported for the MVAR(4), MVAR(9), LSTM(4), and LSTM(9) latent dynamics models. Mean and 10–90% error percentiles are shown *over all the* $C = 20$ cases and $K = \{991, 996\}$ time steps (for width $w = \{9, 4\}$, respectively).

Model	L_1 error, $e_k^{1,(c)}$	L_2 error, $e_k^{2,(c)}$	L_∞ error, $e_k^{\infty,(c)}$
MVAR(4)	0.180 (0.085, 0.246)	0.153 (0.077, 0.211)	0.172 (0.095, 0.242)
MVAR(9)	0.163 (0.087, 0.228)	0.140 (0.079, 0.194)	0.160 (0.092, 0.230)
LSTM(4)	0.177 (0.128, 0.235)	0.156 (0.115, 0.209)	0.185 (0.133, 0.243)
LSTM(9)	0.165 (0.112, 0.223)	0.142 (0.110, 0.190)	0.166 (0.116, 0.227)

Table 5.2, the proposed framework with the use of MVAR models slightly outperforms that of LSTMs in the closed-loop setting across error metrics. The MVAR(9) achieves the lowest errors, indicating accuracy in long-term predictions. The MVAR(4) also performs well, with broader percentiles across test cases. Increasing the lag window from $w = 4$ to $w = 9$ slightly improves MVAR accuracy, demonstrating the benefit of incorporating longer temporal dependencies within the linear autoregressive structure. The LSTM models exhibit slightly higher errors than MVARs, with tighter 10–90% percentiles, while the increase of lag window shows a similar to MVARs improvement.

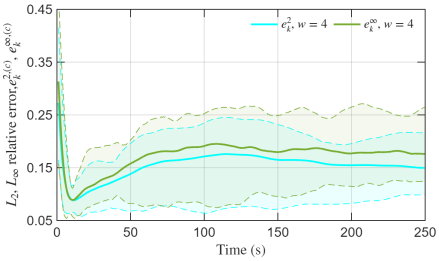
We next assessed the evolution/accumulation of the relative reconstructed errors L_2 and L_∞ over time. Figure 5.4 depicts the average reconstructed relative errors in the ambient–density profile–space, and their 10–90% error percentile ranges across the multiple cases of different, unseen initial conditions in the testing set for all four ROMs. Indicatively, for our illustrations, to provide a comparative "baseline" of the reconstructed error for tiny perturbations of the density profiles, we quantified the sensitivity of the SFM simulations per se with respect to small perturbations of the initial conditions (see in Section 5.4.1). Specifically, we introduced $\pm 1\%$ perturbations in the initial conditions of the density profiles of the reference test cases and computed the relative reconstructed errors between these perturbed initial conditions and their unperturbed counterparts, as shown in Fig. 5.4(e,f).

The time evolution in Fig. 5.4 reveals a characteristic pattern across all models: errors are higher initially, decrease drastically during the first 20 s as the POD reconstruction error baseline diminishes, and then slowly increase towards the end of the simulation due to error accumulation in the recursive forecasting. The MVAR models in Fig. 5.4(a,b) demonstrate remarkable approximation accuracy with the mean relative L_2 error remaining bounded below $\sim$14%(15%) for the MVAR(9)(MVAR(4)) models by the end of the simulation horizon. The respective 10–90% error percentiles are bounded in (7, 25)% errors for the MVAR(4) model, while for the MVAR(9) model they tighten further to (10, 20)% . Both MVAR models achieve errors that compare favorably with the intrinsic system baseline obtained from $\pm 1\%$ perturbed SFM simulations, which shows mean L_2 error of $\sim$ 8% and 10–90% percentiles $(3, 32)\%$ (Fig. 5.4(e,f)).

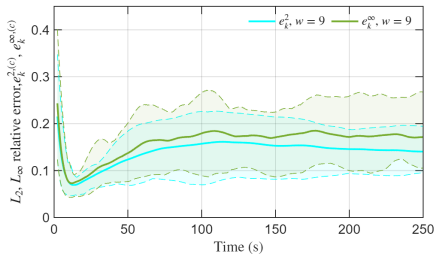
The LSTM models in Fig. 5.4(c,d) provide slightly less accurate predictions than the MVAR models. By the end of the simulation, the LSTM(4) mean relative L_2 error reaches approximately 16%, with 10–90% percentiles widening to $(11, 25)\%$. The LSTM(9) improves marginally, with mean error around 13% and tighter percentiles $(10, 18)\%$. The MVAR and LSTM models demonstrate similar trends, with MVARs providing slightly higher accuracy; a result confirming the averaged in time results reported in Table 5.2.

For qualitative assessment, we directly compare the ground-truth normalized density fields $x^{(c)}(t_k)$ with the corresponding closed-loop predictions $\hat{x}^{(c)}(t_k)$ obtained from the learned evolution operators. Figure 5.5 illustrates representative snapshots produced by the MVAR(9) model, which achieved the best numerical accuracy in the close loop forecast among all ROMs. The predicted densities preserve the dominant spatial structure, propagation direction, and overall mass distribution of the ground truth, indicating that the learned global operator successfully captures the principal macroscopic dynamics over long horizons. The remaining ROM configurations (see Figs. C.2–C.4) exhibit qualitatively similar behavior, albeit with more pronounced discrepancies in the spatial distribution of density. These deviations are particularly visible in regions of steep density gradients.

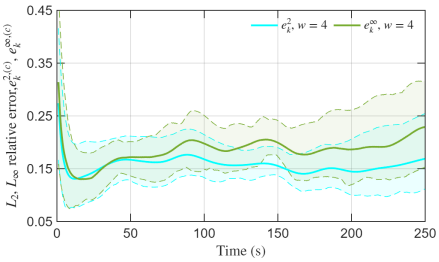
To further quantify spatial accuracy, Fig. C.5 displays the absolute error fields between



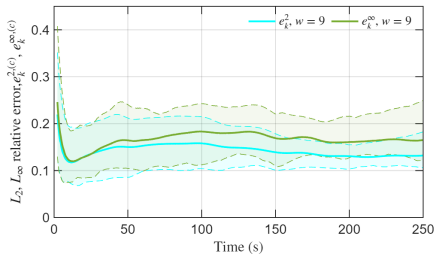
(a) Relative Errors with MVAR(4)



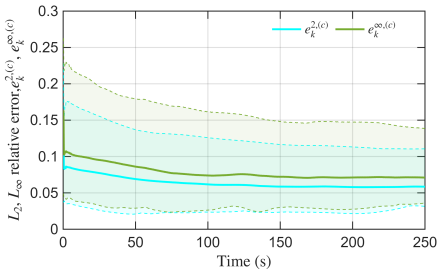
(b) Relative Errors with MVAR(9)



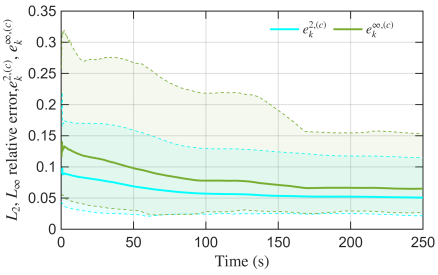
(c) Relative Errors with LSTM(4)



(d) Relative Errors with LSTM(9)



(e) Relative Errors with SFM, +1% perturbation

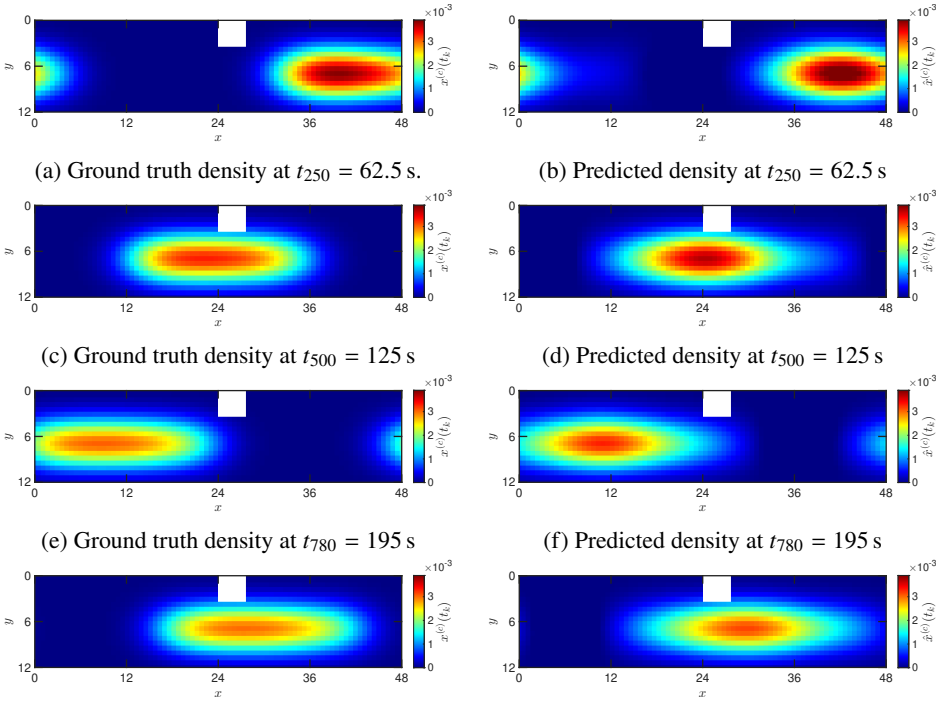


(f) Relative Errors with SFM, −1% perturbation

Figure 5.4: Closed-loop/recursive simulation errors in the ambient–density profile–space, across the testing set over time, using the trained MVAR and LSTM models for the unidirectional flow case. Panels (a)–(d) display the relative L_2 (cyan) and L_∞ (green) errors $e_k^{2,(c)}$ and $e_k^{\infty,(c)}$ in Eq. (5.4.6) for MVAR(4), MVAR(9), LSTM(4) and LSTM(9) models, respectively. Panels (e) and (f) provide a system's "baseline" relative error estimation, obtained from the SFM simulations using ±1% perturbations in the initial conditions for the density profiles (see in Section 5.4.1). Mean relative error (solid) and 10–90% percentiles (dashed) are shown over $C = 20$ cases per time step. For the LSTM models, results correspond to the model achieving the best training performance out of the 50 independent runs.

ground truth and closed-loop predictions for the representative models. All models correctly respect obstacle boundaries, confirming that the structural constraints imposed through the restriction–lifting framework are preserved during forecasting. The dominant error pattern consists of a mild underestimation of peak densities, accompanied by slight overestimation in adjacent low-density regions. This behavior is consistent with projection-based reduced representations, where truncation of higher modes may smooth localized extrema while preserving global mass.

These artifacts are weakest for the higher-order models (MVAR(9) and LSTM(9)) and become more pronounced for the lower-order configurations (MVAR(4) and LSTM(4)). Notably, the magnitude of spatial discrepancies does not accumulate unboundedly over time; instead, error patterns remain spatially structured and gradually attenuate as the flow evolves, as shown in Fig. C.5.



(a) Ground truth density at $t_{250} = 62.5$ s.

(b) Predicted density at $t_{250} = 62.5$ s

(c) Ground truth density at $t_{500} = 125$ s

(d) Predicted density at $t_{500} = 125$ s

(e) Ground truth density at $t_{780} = 195$ s

(f) Predicted density at $t_{780} = 195$ s

(g) Ground truth density at $t_{930} = 232.5$ s

(h) Predicted density at $t_{930} = 232.5$ s

Figure 5.5: Comparison of the "ground truth" $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ normalized densities via the closed-loop time-stepper in Eq. (5.19) for the unidirectional flow case, using the MVAR(9) model for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels (a), (c), (e), and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f), and (h), show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.

Finally, the open-loop (one-step) prediction results are provided in Appendix C; see Table C.1 and Fig. C.1. Similarly to the closed-loop findings above, all four ROMs achieve comparable accuracy in the open-loop setting, with mean L_2 errors of approximately 0.038 across models and narrow 10–90% percentiles. This indicates that while both models capture the local dynamics very accurately in one-step predictions (L_2 errors just above the POD baseline), their long-term forecasting is, in practice, affected by error accumulation that can degrade their performance.

5.5.3 Counterflow case study

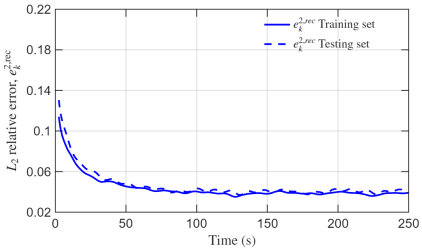
For the second case study, we apply the proposed framework to a counterflow configuration in which two interacting populations are divided equally into groups with opposing flow directions; see Fig. 5.2b. Following the same procedure as in the unidirectional case (implementation details in Sections 5.4.2–5.4.4), the restriction and lifting operators in Eqs. (4.30) and (4.31) are constructed by retaining $d_1 = 6$ and $d_2 = 8$ POD modes for groups $l = 1, 2$, respectively, together with $m = 4$ cross-covariance modes.

The performance of this reduced representation is reported in Fig. 5.6. Panels (a) and (b) show the evolution of the average relative L_2 reconstruction error $e_k^{2,rec}$ for each population over the training and testing datasets, importantly. the reconstruction error rapidly decreases and stabilizes at approximately 3–4% after the first 50 s for both groups, while panels (c)–(e) indicate the selected number of modes satisfying the prescribed energy criterion.

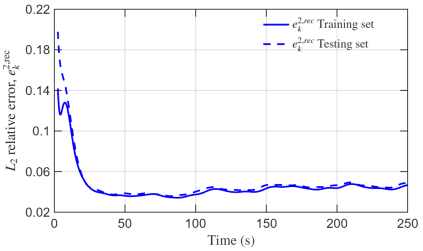
Furthermore, with latent variables $\boldsymbol{y}_c(t_k) = [\boldsymbol{y}_c^{(1)}(t_k), \boldsymbol{y}_c^{(2)}(t_k)] \in \mathbb{R}^{24}$ determined, we trained MVAR and LSTM models following Section 5.4.5. The AIC-identified lag was $w = 10$, and stationarity was verified (ADF test). The MVAR(10) again converged to lower loss function values with substantially less computational time than the LSTM(10) one (see Table 5.3), though both models show deteriorated performance compared to the unidirectional flow case.

Table 5.3: Training results of MVAR(10) and LSTM(10) models at the latent space for the counterflow case. Loss functions $\mathcal{L}_{\text{MVAR}}(\cdot)$ in Eq. (5.9) for the MVAR model and $\mathcal{L}_{\text{LSTM}}(\cdot)$ in Eq. (5.14) for the LSTM model are reported at the end of training, along with the computational times (in seconds) required for training. For the LSTM, the mean and 10–90% error percentiles (in parentheses) are shown across the 50 independent training runs with different random initializations. Additionally, the MSE loss function of the MVAR model in Eq. (5.18) is reported for direct comparison to the MSE-based loss $\mathcal{L}_{\text{LSTM}}(\cdot)$.

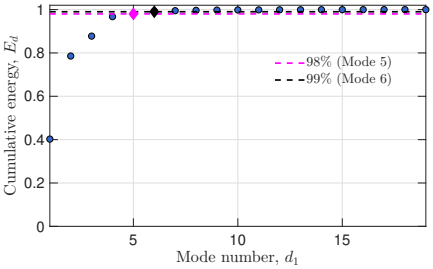
Model	$\mathcal{L}_{\text{MVAR}}(\cdot)$	$MSE_{\text{MVAR}}(\cdot)$	$\mathcal{L}_{\text{LSTM}}(\cdot)$	**Comput. Time (s)**
MVAR(10)	5.096×10^{-6}	1.07×10^{-11}	-	0.031
LSTM(10)	-	-	$1.44\ (1.38, 1.71) \times 10^{-7}$	104 (84, 226)



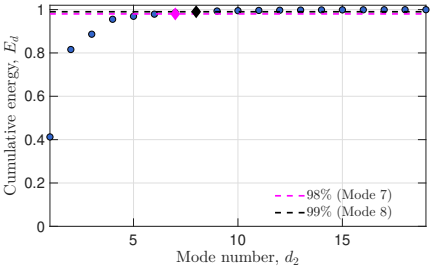
(a) Augmented POD reconstruction error (group 1)



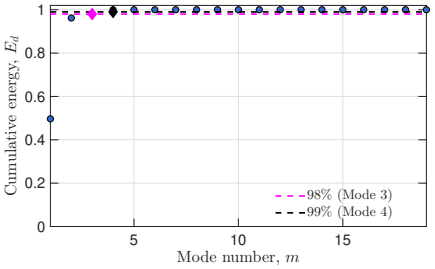
(b) Augmented POD reconstruction error (group 2)



(c) POD cumulative energy (group 1)



(d) POD cumulative energy (group 2)



(e) m cross-covariance modes retained.

Figure 5.6: Accuracy of the restriction and lifting operators in Eqs. (4.30) and (4.31) for the counterflow case. Panels (a) and (b) depict the average (over $C = 20$ cases with different initial conditions) reconstruction errors $e_k^{2,rec}$ in Eq. (4.7) for groups 1 and 2, respectively, in time for the training and testing datasets. The augmented basis in Eq. (2) is constructed with $d_1 = 6$, $d_2 = 8$ POD modes retained for the group-specific matrices $X_{tr}^{(1)}$, $X_{tr}^{(2)}$ and $m = 4$ modes retained for cross-covariance matrix $C^{(12)}$. Panels (c)–(e) show the minimum number of POD modes (indicated by colored diamonds) required to retain the prescribed percentage of energy in the training dataset, for groups $l = 1, 2$ and for the m cross-covariance modes.

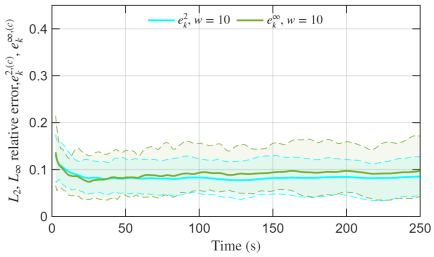
We now evaluate the end-to-end performance of the proposed framework in the ambient–density profile–space in long-term (closed-loop) predictions. Following the same quantification procedure as in Section 5.5.1, we again observe that the closed-loop prediction accuracy with the use of MVAR is superior to that of LSTM (see Table 5.4). In particular, the mean, overall testing cases and time steps, reconstructed L_2 error of the MVAR(10) model is 8% with tight 10–90% percentiles, while that of the LSTM(10) model is $\sim$ 10% with wider percentiles for both population groups.

Table 5.4: Closed-loop (recursive) prediction errors in the ambient density profile space for the testing set using the trained MVAR and LSTM (best out of 50 training runs) models for the counterflow case. The relative reconstructed L_1, L_2, and L_∞ errors $e_k^{1,(c)}$, $e_k^{2,(c)}$, and $e_k^{\infty,(c)}$ in Eq. (5.4.6) are reported separately for group 1 and group 2 using the MVAR(10) and LSTM(10) latent dynamics models. Mean values and 10–90% error percentiles (in parentheses) are shown over all $C = 20$ cases and $K = 990$ time steps.

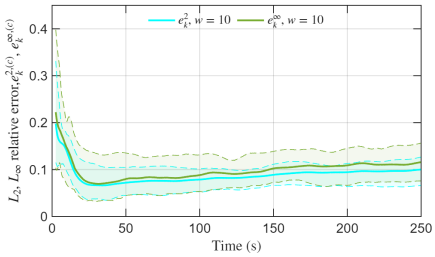
Model	L_1 error	L_2 error	L_∞ error
group 1			
MVAR(10)	0.108 (0.064, 0.164)	0.083 (0.046, 0.123)	0.091 (0.046, 0.151)
LSTM(10)	0.132 (0.090, 0.201)	0.104 (0.067, 0.168)	0.113 (0.061, 0.197)
group 2			
MVAR(10)	0.118 (0.086, 0.152)	0.088 (0.058, 0.115)	0.099 (0.060, 0.144)
LSTM(10)	0.123 (0.095, 0.157)	0.095 (0.068, 0.123)	0.115 (0.070, 0.152)

The detailed error evolution over time is displayed in Fig. 5.7. The MVAR(10) model shows mean L_2 errors dropping to $\sim 9-10\%$ within the first 20 s for both population groups. These errors remain bounded over the long-time horizon at values slightly over 10%, exhibiting high accuracy in comparison to the baseline accuracy of pure restriction and lifting operators ($3-4\%$ after the first 50 s for both groups; see Fig. 5.6). In contrast, LSTM(10) exhibits slightly larger errors that stabilize at $13-14\%$ by the simulation end. Across both models, group 2 shows slightly higher errors and wider uncertainty bands.

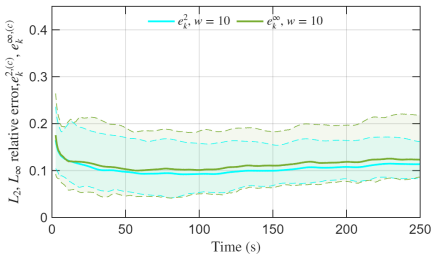
Additionally, we provide a direct comparison of the ground truth densities with their closed-loop predictions for both groups (see Figs. 5.8 and 5.9 in D). Both MVAR and LSTM models accurately reproduce the two dominant density lobes and capture migration and alignment patterns of the two interacting groups. These results, together with the spatial distribution of the absolute errors between ground truth and closed-loop predictions (see Fig. D.2), further show that both models respect obstacle regions, without significant errors accumulating in the immediate obstacle proximity, while accurately reproducing localized congestion zones formed upstream and downstream of the obstacle.



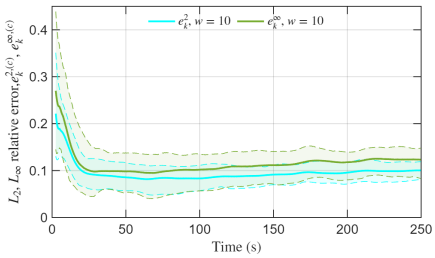
(a) Relative Errors with MVAR(10) for group 1



(b) Relative Errors with MVAR(10) for group 2



(c) Relative Errors with LSTM(10) for group 1

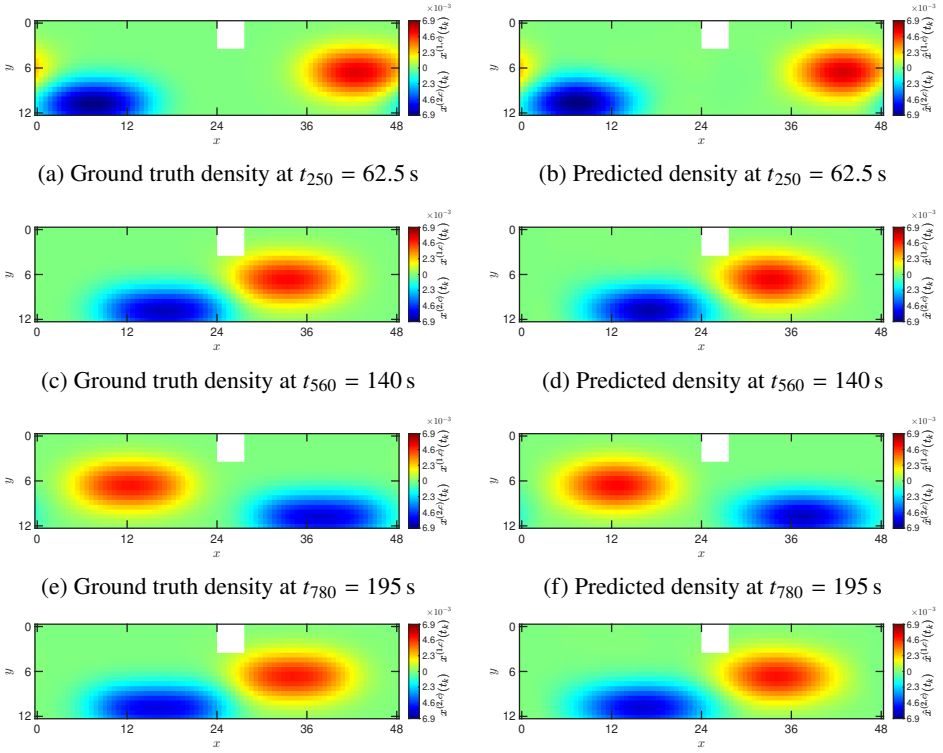


(d) Relative Errors with LSTM(10) for group 2

Figure 5.7: Closed-loop/recursive simulation errors in the ambient–density profile–space, across the testing set over time, using the MVAR and LSTM models for the counterflow case. Panels (a,b) and (c,d) display the relative reconstructed L_2 (cyan) and L_∞ (green) errors $e_k^{2,(c)}$ and $e_k^{\infty,(c)}$ in Eq. (5.4.6) for MVAR(10) and LSTM(10) models, respectively. The left and right column panels show errors for group 1 and 2, respectively. Mean relative errors (solid) and 10–90% error percentiles (dashed) are reported over $C = 20$ cases per time step. For the LSTM model, results correspond to the run achieving the best training performance out of the 50 independent initializations.

Minor discrepancies include peak attenuation and spatial dispersion of the Gaussian-like lobe structure of each group, which are slightly more pronounced with the LSTM(10) model, especially as time progresses; a finding that is consistent with LSTM's higher error metrics compared to MVAR in Fig. 5.7.

For completeness, we also provide the open-loop/one-step prediction results in Table D.1 and Fig. D.1 of Appendix D. As in the unidirectional flow case, both MVAR and LSTM models achieve comparable accuracy with mean L_2 errors $\sim 4-6\%$ and very narrow 10–90% percentiles. This very accurate performance in one-step predictions demonstrates once again the effect of error accumulation introduced in the recursive predictions for long-term horizons.



(a) Ground truth density at $t_{250} = 62.5$ s

(b) Predicted density at $t_{250} = 62.5$ s

(c) Ground truth density at $t_{560} = 140$ s

(d) Predicted density at $t_{560} = 140$ s

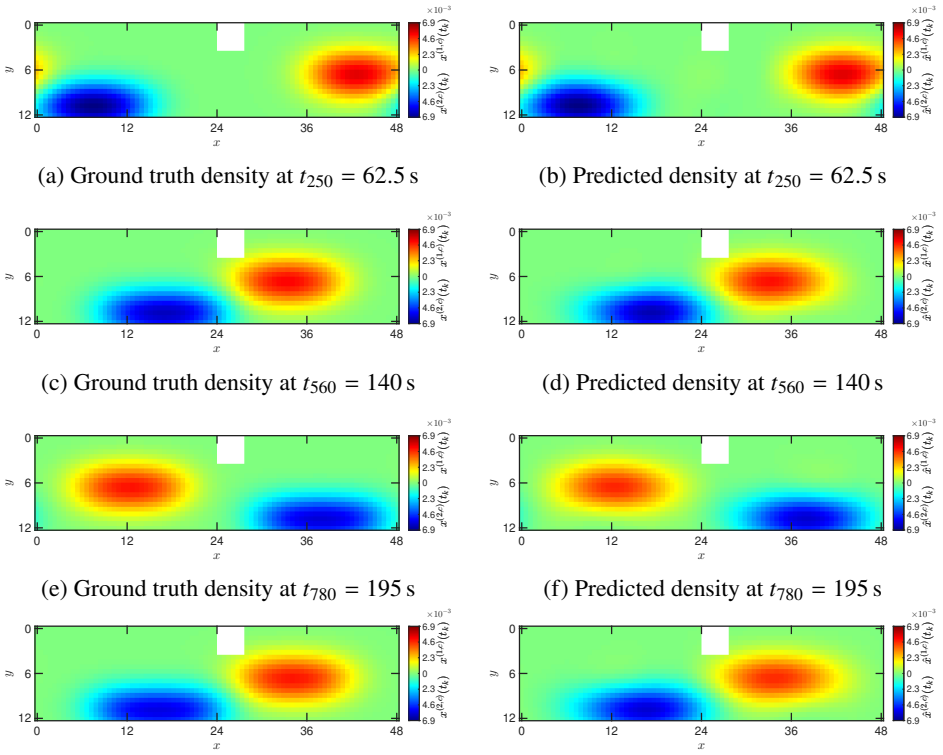
(e) Ground truth density at $t_{780} = 195$ s

(f) Predicted density at $t_{780} = 195$ s

(g) Ground truth density at $t_{950} = 237.5$ s

(h) Predicted density at $t_{950} = 237.5$ s

Figure 5.8: Comparison of the "ground truth" $x^{(l,c)}(t_k)$ and the predicted $\hat{x}^{(l,c)}(t_k)$ normalized densities via the closed-loop time-stepper in Eq. (5.19) for groups $l = 1, 2$ in the counterflow case, using the MVAR(10) model for an unseen case of the testing set; initialization at the 15th row of Table B.2. Panels (a), (c), (e) and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f) and (h) show the predicted one, for the timesteps $t_{250} = 62.5$ s, $t_{560} = 140$ s, $t_{780} = 195$ s, and $t_{950} = 237.5$ s, respectively. The group-specific densities are superimposed with the blue and red lobes corresponding to groups 1 and 2, respectively.



(a) Ground truth density at $t_{250} = 62.5$ s

(b) Predicted density at $t_{250} = 62.5$ s

(c) Ground truth density at $t_{560} = 140$ s

(d) Predicted density at $t_{560} = 140$ s

(e) Ground truth density at $t_{780} = 195$ s

(f) Predicted density at $t_{780} = 195$ s

(g) Ground truth density at $t_{950} = 237.5$ s

(h) Predicted density at $t_{950} = 237.5$ s

Figure 5.9: Comparison of the "ground truth" $x^{(l,c)}(t_k)$ and the predicted $\hat{x}^{(l,c)}(t_k)$ normalized densities via the closed-loop time-stepper in Eq. (5.19) for groups $l = 1, 2$ in the counterflow case, using the LSTM(10) model for an unseen case of the testing set; initialization at the 15th row of Table B.2. Panels (a), (c), (e) and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f) and (h) show the predicted one, for the timesteps $t_{250} = 62.5$ s, $t_{560} = 140$ s, $t_{780} = 195$ s, and $t_{950} = 237.5$ s, respectively. The group-specific densities are superimposed with the blue and red lobes corresponding to groups 1 and 2, respectively.

Finally, the proposed framework achieves higher accuracy in the counterflow case than in the unidirectional flow case, and this difference is consistent across ROM model selections. This may be attributed to the larger latent dimension (24 vs 6), which provides a richer representation space. However, this comes at increased computational cost, highlighting the accuracy-efficiency trade-off inherent in dimensionality selection.

5.5.4 MVAR vs. LSTM performance

The comparison between model classes in both unidirectional and counterflow case studies reveals that the use of MVARs provides a slightly more accurate and reliable framework for long-term predictions than that of LSTMs. This is clearly demonstrated by their lower errors and narrower error percentiles in the closed-loop setting (Table 5.2, Fig. 5.4 and Table 5.4, Fig. 5.7). While all models perform remarkably well in an open-loop setting (see Table C.1, Fig. C.1 and Table 5.4, Fig. D.1), the effect of error accumulation in recursive predictions degrades their predictive accuracy. For instance, in the unidirectional flow case the MVAR(4) model's mean L_2 error across time increased from ∼4% (open-loop) to ∼15% (closed-loop), and the LSTM(4) model's error increased from ∼4% to ∼16%. This consistent pattern in both case studies with different latent dimensions (6 vs 24) and interaction complexities is attributed to the challenge of error accumulation in closed-loop forecasting; the linear structure of the MVAR model proves to be less susceptible to this instability than complex, nonlinear LSTMs. While LSTMs can theoretically capture richer nonlinear dynamics, their performance degrades due to the "curse of dimensionality" in their training, i.e., finding a global optimum, accompanied by a distribution shift [176, 177] (i.e., the error accumulation of the recursive scheme during inference, that shifts the distribution upon which the models were trained), despite their ability to learn the one-step dynamics effectively. In addition, there are several challenges regarding the "correct" initialization of LSTMs [178], especially in dynamical systems and multiscale modeling tasks. In contrast, the training of the linear MVAR models is based on least squares regression, thus giving a global optimum. For the particular applications, the results suggest that MVARs are a much better choice for long-term horizon predictions in the latent space. This is a result in line also with other studies (see e.g., [114]).

The proposed framework offers a balanced trade-off between prediction accuracy and computational efficiency for coarse-grained crowd dynamics. For the unidirectional flow case (250 s horizon, $K = 1000$ steps), one conventional simulation requires 80 s for SFM integration plus approximately 18 s for density field extraction via KDE, totaling 98 s per complete prediction. Our framework's online closed-loop prediction requires $w \times (t_{ID} + t_R)$ s for density extraction and restriction of the first w lagged snapshots, followed by $(1000 - w) \times (t_F + t_L)$ for per-step ROM forecasting and lifting operations; see Eq. (5.19). This results in total online execution times of 0.397, 0.887, 1.413, and 1.871 s for the MVAR(4), MVAR(9), LSTM(4), and LSTM(9) models, respectively, corresponding to speed-ups of 247×, 110×, 69×, and 52× relative to the conventional approach. For the counterflow case, one conventional simulation requires a total of 77.35 s, while our framework yields total online execution times of 0.781 and 2.141 s for

the MVAR(10) and LSTM(10) models, respectively, corresponding to speed-ups of 99× and 36×. These online times include lifting at every time step for a fair comparison with the conventional simulation output. In practice, with on-demand lifting (i.e., applying $\mathcal{L}$ only when density fields are needed), the speed-ups would be even greater, as the framework evolves in the latent space (Eq. (5.19)).

Offline costs (one-time) for the unidirectional flow case include $(80 + 18) \times 20 = 1960$ s for generating 20 training trajectories via SFM with KDE extraction, 4.24 s for POD basis computation, and model training times of 0.0013, 0.0017, 62, and 86 s for MVAR(4), MVAR(9), LSTM(4), and LSTM(9) models respectively (the LSTMs times are averaged over 50 runs; see Table 5.1). For the counterflow case, the offline costs are 1547 s for generating 20 training trajectories, 15.51 s for the augmented POD basis, and model training times of 0.031 and 104 s for MVAR(10) and LSTM(10) models respectively (see Table 5.3). While offline costs appear significant, they become negligible when amortized over multiple predictions. For example, the total offline cost for LSTM(9) in the unidirectional flow case (2050.24 s) equals just 21 conventional simulations; a cost quickly recouped when dozens or hundreds of predictions are required, as in parameter studies or real-time applications.

Overall, MVAR variants prove particularly efficient for real-time applications, while maintaining better accuracy than their LSTM counterparts. These gains are especially valuable for real-world scenarios, where the number of pedestrians renders the conventional approaches infeasible.

5.6 Discussion

Chapters 4 and 5 have established a coherent data-drive framework for modeling macroscopic crowd dynamics. In Chapter 4, we introduced a reduced-order formulation in which spatial density fields are treated as high-dimensional macroscopic states. Through the construction of Restriction and Lifting operators based on POD, we defined reduced latent coordinates that preserve mass by construction and admit a consistent interpretation as macroscopic state variables.

Building on this foundation, Chapter 5 introduced the temporal component of the framework by learning discrete-time global evolution operators directly in the reduced latent space. In contrast to classical EF framework, which rely on local coarse time-steppers constructed on demand, the present approach identifies a global surrogate operator acting on latent states. This operator is learned from data generated by microscopic simulations, yet it evolves macroscopic states autonomously without recourse to further particle-level integration. The resulting formulation therefore provides a fully data-driven approximation of the effective solution operator of the underlying, unknown macroscopic PDE.

The numerical investigations across both unidirectional and counterflow configurations demonstrate that the proposed framework achieves accurate long-horizon forecasting

while preserving physical consistency. In particular, the restriction–lifting construction ensures exact mass conservation, and the learned latent operators remain stable under recursive (closed-loop) prediction. Notably, linear MVAR models in the carefully constructed latent space consistently match or outperform more complex nonlinear LSTM models in long-horizon forecasting, highlighting the importance of representation quality over model complexity. Furthermore, the computational gains relative to conventional microscopic simulations confirm the suitability of the approach for real-time and large-scale applications.

Taken together, Chapters 3 and 5 establish the following key result: macroscopic crowd dynamics can be treated as a reduced-order state-space system, whose evolution operator can be learned globally in latent coordinates while maintaining structural physical constraints. The framework thus bridges microscopic simulation data and macroscopic dynamical systems modeling without postulating or identifying explicit partial differential equations.

Having constructed and validated the reduced macroscopic evolution operator, we are now positioned to move beyond forecasting toward system-level tasks. Once crowd density fields are represented as states in a reduced dynamical system, the resulting reduced-order model enables the application of classical tools from control theory and state estimation. This observation motivates Part III of the thesis, where the learned operator framework is extended to develop physics-informed machine learning formulations for state estimation (observer design) and control.

Part III

Estimation and Control

6 State estimation for latent macroscopic crowd dynamics

Abstract Chapter 5 established discrete-time macroscopic evolution operators governing crowd dynamics in an appropriate latent space. In realistic monitoring and control framework, however, only partial observations of these latent variables are typically available. In this chapter, the reduced macroscopic model is interpreted as a nonlinear discrete-time state-space system and the problem of latent state reconstruction is formulated within a rigorous observer framework. Exact nonlinear observer linearization is employed to construct a full-order observer with assignable convergence properties. The design requires the solution of a system of nonlinear functional equations defining a coordinate transformation that maps the learned nonlinear dynamics into linear form. To compute this transformation in moderately high-dimensional latent spaces, a physics-informed neural network (PINN) formulation is introduced, enforcing the functional relations, equilibrium invariance, and local linearization constraints. The methodology is first validated on benchmark systems with analytically known dynamical systems and associated nonlinear transformations and subsequently applied to the LSTM-based evolution operator. Numerical experiments demonstrate stable reconstruction of latent states under partial observability and physically consistent recovery of macroscopic density fields.

6.1 Motivation

The macroscopic evolution framework developed in Chapters 4 and 5 provides a discrete-time operator acting on reduced latent representations of crowd density fields. In particular, the POD basis defines restriction and lifting operators that map between the high-dimensional density space and a reduced latent coordinate vector, previously denoted by $\boldsymbol{y}_c(t_k) \in \mathbb{R}^d$, whose temporal evolution is governed by a learned nonlinear operator. The construction of this operator assumes *full access* to the latent state vector. In practice, however, such complete information is rarely available. Experimental measurements, camera-based sensing systems, or embedded monitoring infrastructures typically provide only partial information about the macroscopic density field [179]. When projected onto the reduced POD coordinates, this corresponds to observing only a subset of the latent coefficients.

To facilitate the connection with nonlinear observer theory, we now adopt standard state-space notation. The latent POD coefficients previously written as $\boldsymbol{y}_c(t_k)$ will henceforth be denoted by

$$\mathbf{x}(t_k) \in \mathbb{R}^d, \tag{6.1}$$

while the observable subset of coefficients will be denoted by

$$\mathbf{y}(t_k) \in \mathbb{R}^p, \qquad p < d. \tag{6.2}$$

This change of notation does not modify the learned evolution operator; it merely embeds the reduced macroscopic dynamics into a conventional nonlinear state-space representation suitable for observer analysis. Under this notation, the learned reduced-order model can be written in the canonical discrete-time form

$$\begin{aligned} \mathbf{x}(t_{k+1}) &= \Phi(\mathbf{x}(t_k)), \\ \mathbf{y}(t_k) &= h(\mathbf{x}(t_k)), \end{aligned} \tag{6.3}$$

where $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ is the nonlinear evolution map realized by the trained reduced-order model, and $h : \mathbb{R}^d \to \mathbb{R}^p$ represents the observation operator selecting the measurable latent components.

The presence of partial observability introduces a fundamental limitation. The learned macroscopic operator Φ requires the full latent state $\mathbf{x}(t_k)$ for forward evolution. If certain coefficients are missing, the operator cannot be applied directly, and lifting back to the physical density field becomes inconsistent. Consequently, the reduced macroscopic model becomes operational only if the unobserved latent variables can be systematically reconstructed from the available measurements.

This issue is not merely technical as the latent POD coefficients encode coherent spatial structures of the crowd density field, including dominant flow modes and obstacle-induced deflections. Missing components may correspond to dynamically significant

structures whose absence leads to physically inaccurate predictions. Accurate reconstruction of these coefficients is therefore essential to ensure consistent macroscopic evolution.

An important structural property required for state reconstruction in crowd dynamics is the existence of a physically meaningful equilibrium. In this thesis, this requirement is satisfied because the density fields are centered prior to POD construction, and the underlying crowd dynamics preserve spatially uniform density states. The origin $\mathbf{x} = \mathbf{0}$ therefore corresponds to a uniform equilibrium configuration. Accordingly, the learned operator satisfies

$$\Phi(\mathbf{0}) = \mathbf{0}, \tag{6.4}$$

ensuring that the origin is a genuine equilibrium of the reduced dynamics rather than a coordinate artifact. The reconstruction of the full latent state $\mathbf{x}(t_k)$ from partial measurements $\mathbf{y}(t_k)$ thus constitutes a nonlinear state estimation problem for a discrete-time dynamical system with known equilibrium structure. Linear reconstruction strategies are generally insufficient in this setting, as the learned operator Φ is nonlinear and obtained directly from data. A principled solution must therefore respect the nonlinear structure of the dynamics while providing guarantees on stability and convergence of the reconstructed state.

In the following sections, we address this problem through exact nonlinear observer linearization, which enables the systematic construction of observers with assignable convergence properties and embeds the learned macroscopic operator within a rigorous state estimation framework combined with physics-informed machine learning.

6.2 Exact nonlinear observer linearization

Following the nonlinear state-space formulation introduced in Section 6.1, we now present the exact nonlinear observer linearization framework for discrete-time nonlinear systems. This methodology provides the structural foundation for nonlinear state reconstruction and will subsequently be applied to the learned macroscopic evolution operator.

6.2.1 Preliminaries

We consider nonlinear discrete-time autonomous dynamical systems admitting the state-space representation introduced by Eq. (6.3): The nonlinear map $\Phi(\mathbf{x})$ is assumed real analytic on $\mathbb{R}^d$, and $h(\mathbf{x})$ is real analytic on $\mathbb{R}^d$. Without loss of generality, the origin $\mathbf{x} = \mathbf{0}$ is assumed to be an equilibrium point of Eq. (6.3), i.e., $\Phi(\mathbf{0}) = \mathbf{0}$ and $h(\mathbf{0}) = \mathbf{0}$. In the case of a non-zero equilibrium $\mathbf{x}_0$ satisfying $\Phi(\mathbf{x}_0) = \mathbf{x}_0$, the coordinate transformation $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ yields a system with equilibrium at the origin. The following assumptions are imposed:

Assumption 6.2.1. The Jacobian matrix F of $\Phi(\mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{0}$, namely

$$F = \left(\frac{\partial \Phi}{\partial \mathbf{x}}\right)(\mathbf{0}), \tag{6.5}$$

has eigenvalues f_i that all lie in the Poincaré domain.

Assumption 6.2.2. Denoting by H the $p \times d$ matrix

$$H = \left(\frac{\partial h}{\partial \mathbf{x}}\right)(\mathbf{0}), \tag{6.6}$$

it is assumed that the following $d \times d$ matrix O:

$$O = \begin{bmatrix} H \\ HF \\ \vdots \\ HF^{d-1} \end{bmatrix} \tag{6.7}$$

has rank d. This assumption states that the system in Eq. (6.3) is locally observable around the origin $\mathbf{x} = \mathbf{0}$.

6.2.2 Observer Definition

A nonlinear discrete-time state observer driven by the available measurements $\mathbf{y}(t_k)$ and capable of reconstructing the state vector of system (6.3) conforms to the following structure:

Definition 6.2.1. A dynamical system

$$\mathbf{z}(t_{k+1}) = \Psi(\mathbf{z}(t_k), \mathbf{y}(t_k)) \tag{6.8}$$

with $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^p$, and $\Psi : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$, is called a full-order observer for Eq. (6.3), if there exists a locally invertible operator $T(\mathbf{x})$ with $T : \mathbb{R}^d \to \mathbb{R}^d$ and $T(\mathbf{0}) = \mathbf{0}$, such that if $\mathbf{z}(0) = T(\mathbf{x}(0))$ with $\mathbf{x}(0)$ sufficiently close to $\mathbf{0}$, then $\mathbf{z}(t_k) = T(\mathbf{x}(t_k))$ for all $t_k > 0$.

Definition 6.2.1 implies that if Ψ and T are related through the following system of functional equations:

$$\begin{aligned} T(\Phi(\mathbf{x})) &= \Psi(T(\mathbf{x}), h(\mathbf{x})) \\ T(\mathbf{0}) &= \mathbf{0}, \end{aligned} \tag{6.9}$$

then the $\mathbf{y}$-driven dynamical system given by Eq. (6.8) represents an observer for system in Eq. (6.3), whenever the operator $T(\mathbf{x})$ is invertible. It is important to note that the initial condition $T(\mathbf{0}) = \mathbf{0}$ reflects the preservation of equilibrium properties under the proposed state transformation.

In the special case of an identity observer, condition (6.9) reduces to

$$\Phi(\mathbf{x}) = \Psi(\mathbf{x}, h(\mathbf{x})). \tag{6.10}$$

Condition (6.10) expresses the well-known requirement entailed by the standard definition of a nonlinear observer.

6.2.3 Single-step exact observer linearization

The vector function $\Psi(\mathbf{z}, \mathbf{y})$ in Eq. (6.8) can be arbitrarily selected, provided that the system of functional equations (6.9) admits an invertible solution $T(\mathbf{x})$. Furthermore, stability requirements can be imposed at the observer design stage, where Ψ is selected *a priori* to ensure asymptotic stability.

For simplicity, linear observer dynamics in the transformed states $\mathbf{z}$ up to an output injection term are requested by choosing:

$$\Psi(\mathbf{z}, \mathbf{y}) = A\mathbf{z} + b(\mathbf{y}), \tag{6.11}$$

where A is a constant matrix and b a vector function defined on $\mathbb{R}^p$. Under such a requirement, the operator $T(\mathbf{x})$ ought to satisfy the following system of first-order non-homogeneous functional equations:

$$\begin{aligned} T(\Phi(\mathbf{x})) &= AT(\mathbf{x}) + b(h(\mathbf{x})) \\ T(\mathbf{0}) &= \mathbf{0}, \end{aligned} \tag{6.12}$$

with $T : \mathbb{R}^d \longrightarrow \mathbb{R}^d$ being the solution of (6.12).

6.2.4 Existence theorem

Theorem 6.2.1. Suppose that Assumptions 6.2.1 and 6.2.2 hold true. Let $B = \left(\frac{\partial b}{\partial y}\right)(0)$ and assume that the pair of matrices $\{A, B\}$ is chosen to be controllable and the eigenvalues $f_i, (i = 1, ..., d)$ of matrix $F = \left(\frac{\partial \Phi}{\partial x}(0)\right)$ are not related to the eigenvalues $\lambda_i, (i = 1, ..., d)$ of matrix A through any equation of the form:

$$\prod_{i=1}^{d} k_i^{m_i} = \lambda_j \tag{6.13}$$

$(j = 1, ..., d)$, where all the m_i's are non-negative integers that satisfy:

$$\sum_{i=1}^{d} m_i > 0. \tag{6.14}$$

Then, there exists a unique analytic and locally invertible solution $T(\mathbf{x})$ to the system of functional equations (6.12) as well as a nonlinear map $\mathbf{z} = T(\mathbf{x})$ that makes the dynamical system:

$$\mathbf{z}(t_{k+1}) = A\mathbf{z}(t_k) + b(\mathbf{y}(t_k)) \tag{6.15}$$

an observer for system (6.3) in the sense of Definition 6.2.1.

6.2.5 Observer in original coordinates and error dynamics

The proposed nonlinear discrete-time state observer expressed in the original state variables attains the following form:

$$\hat{\mathbf{x}}(t_{k+1}) = T^{-1}\left[T(\Phi(\hat{\mathbf{x}}(t_k))) + b(\mathbf{y}(t_k)) - b(h(\hat{\mathbf{x}}(t_k)))\right] \tag{6.16}$$

where $\hat{\mathbf{x}} \in \mathbb{R}^d$ is the state estimate and $T(\mathbf{x})$, $T : \mathbb{R}^d \longrightarrow \mathbb{R}^d$ is the operator constructed by the solution to the associated system of functional equations (6.12). The structure of the above observer is mathematically realized through the following two terms:

(i) $\Phi(\hat{\mathbf{x}})$ is a replica of the system dynamics (6.3), and

(ii) $b(\mathbf{y}(t_k)) - b(h(\hat{\mathbf{x}}(t_k)))$ is a feedback term accounting for the mismatch between the actual sensor measurement $\mathbf{y}(t_k)$ and its estimate $h(\hat{x}(t_k))$.

Furthermore, the proposed state observer in Eq. (6.15) induces linear estimation error dynamics with assignable dynamic modes in the transformed variables $\mathbf{z} = T(\mathbf{x})$:

$$\begin{aligned}
\mathbf{e}_z(t_{k+1}) = \mathbf{z}(t_{k+1}) - \hat{\mathbf{z}}(t_{k+1}) = T(\mathbf{x}(t_{k+1})) - T(\hat{\mathbf{x}}(t_{k+1})) = \\
= T(\Phi(\mathbf{x}(t_k))) - T(\Phi(\hat{\mathbf{x}}(t_k))) - b(\mathbf{y}(t_k)) + b(h(\hat{\mathbf{x}}(t_k))) = \\
= AT(\mathbf{x}(t_k)) + b(h(\mathbf{x}(t_k))) - AT(\hat{\mathbf{x}}(t_k)) - b(h(\hat{\mathbf{x}}(t_k))) - b(\mathbf{y}(t_k)) + b(h(\hat{\mathbf{x}}(t_k))) = \\
= AT(\mathbf{x}(t_k)) - AT(\hat{\mathbf{x}}(t_k)) = A(\mathbf{z}(t_k) - \hat{\mathbf{z}}(t_k)) = A\mathbf{e}_z(t_k).
\end{aligned} \tag{6.17}$$

Therefore, if the matrix A is chosen with an eigenspectrum ensuring stability (i.e., eigenvalues located strictly inside the unit disc), the magnitude of these eigenvalues directly regulates the rate of decay of the estimation error to zero. Local invertibility of the transformation operator $T(\mathbf{x})$ implies that the state estimate $\hat{x}$ asymptotically approaches the actual state x.

It is worth noting that the choice of the matrix pair $\{A, B\}$ reflects the design degrees of freedom inherent in the proposed observer. The controllability condition ensures existence and uniqueness of the linearizing transformation operator $T(\mathbf{x})$, whereas the eigenspectrum of matrix A enables assignment of the desired convergence rate.

6.2.6 Comparison with two-step approaches

Finally, it should be pointed out, that the above exact observer linearization approach is fundamentally different from the ones presented in other studies (see e.g., in [180, 181, 182]) where as a first step, a nonlinear coordinate transformation that transforms the original system (6.3) into a linear one (up to an output injection term) with a linear output map is sought:

$$\begin{aligned} \mathbf{z}(t_{k+1}) &= A\mathbf{z}(t_k) + b(\mathbf{y}(t_k)) \\ \mathbf{y}(t_k) &= c\mathbf{z}(t_k) \end{aligned} \tag{6.18}$$

The design of the observer in [180, 181, 182] is then completed in a second step where a standard linear Luenberger observer is used for the transformed system (6.18):

$$\hat{\mathbf{z}}(t_{k+1}) = A\hat{\mathbf{z}}(t_k) - L(\mathbf{y} - c\hat{\mathbf{z}}(t_k)) + b(\mathbf{y}(t_k)), \tag{6.19}$$

with L being the Luenberger observer gain and $\hat{z}$ the state estimate in the transformed coordinates.The estimate $\hat{\mathbf{x}}$ of the original state vector $\mathbf{x}$ is then recovered, through the inverse transformation employed in the first step of the design procedure. Please notice that the requirement of a linear output map in the transformed system (6.18) introduced in the above two-step approach, represents the main mathematical reason for the emergence of quite restrictive conditions in [180, 181, 182]. Within the proposed exact observer linearization framework however, the state observer is a dynamical system that is driven by the measured output variable and designed through an appropriate coordinate transformation in a single step without the redundant requirement of linearity of the output map imposed on the transformed linear system (6.18).

6.3 Numerical solution of the functional equations

A practical implementation of the proposed nonlinear discrete-time observer design method requires the development of a solution scheme for the nonlinear functional equations (6.12).

6.3.1 Taylor series-based recursive construction

As mentioned earlier, the functions $\Phi(\mathbf{x})$, $h(\mathbf{x})$, as well as the transformation operator $T(\mathbf{x})$ are all locally analytic. Therefore, as suggested in [183], a solution method can be based on a multivariate Taylor series expansion of $\Phi(\mathbf{x})$, $h(\mathbf{x})$ and $T(\mathbf{x})$, followed by a procedure that equates same-order Taylor coefficients of both sides of the functional equations (6.12).

Recursive algebraic formulas are thus generated that are linear with respect to the Taylor coefficients of the unknown solution. In particular, the N-th order Taylor coefficients of $T(\mathbf{x})$ can be calculated given those up to order $N-1$ obtained in previous recursive steps.

Using tensorial notation:

a) Entries of matrix A are denoted by a_i^j.

b) Partial derivatives of component $f_\mu(\mathbf{x})$ at $x = 0$ are:

$$\begin{aligned} f_\mu^i &= \frac{\partial f_\mu}{\partial x_i}(0) \\ f_\mu^{ij} &= \frac{\partial^2 f_\mu}{\partial x_i \partial x_j}(0) \\ f_\mu^{ijk} &= \frac{\partial^3 f_\mu}{\partial x_i \partial x_j \partial x_k}(0) \end{aligned} \tag{6.20}$$

c) Standard summation convention is used.

The l-th component $T_l(\mathbf{x})$ is expanded as:

$$\begin{aligned} T_l(\mathbf{x}) &= \frac{1}{1!} T_l^{i_1} x_{i_1} + \frac{1}{2!} T_l^{i_1 i_2} x_{i_1} x_{i_2} + \ldots \\ &+ \frac{1}{N!} T_l^{i_1 \ldots i_N} x_{i_1} \ldots x_{i_N} + \ldots \end{aligned} \tag{6.21}$$

Matching coefficients yields:

$$\sum_{L=1}^{N} \sum_{\substack{0 \le m_1 \le \cdots \le m_L \\ m_1 + \cdots + m_L = N}} T_l^{j_1 \cdots j_L} \Phi_{j_1}^{m_1} \cdots \Phi_{j_L}^{m_L} = a_l^\mu T_\mu^{i_1 \cdots i_N} + b_l^\mu h_\mu^{i_1 \cdots i_N} \tag{6.22}$$

where $i_1, \ldots, i_N = 1, \ldots, d$ and $l = 1, \ldots, d$.

These recursive relationships define linear algebraic equations in the unknown coefficients $T_\mu^{i_1, \ldots, i_N}$. However, symbolic Taylor-based solutions become computationally prohibitive in medium-scale dimensional systems or in the presence of steep gradients.

Linearization at the equilibrium

The linearization of Eq. (6.3) around the equilibrium $(\mathbf{0}, \mathbf{0})$ gives:

$$\mathrm{d}\mathbf{x}(t_{k+1}) = \frac{\partial \Phi}{\partial \mathbf{x}}(\mathbf{0})\, \mathrm{d}\mathbf{x}(t_k); \tag{6.23}$$

On the other hand, since $\Phi(\mathbf{x}(t_k)) = \mathbf{x}(t_{k+1})$, the linearization of the transformed system (6.12) around the equilibrium reads:

$$\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\, \mathrm{d}\mathbf{x}(t_{k+1}) = \left(A \frac{\partial T}{\partial \mathbf{x}}(\mathbf{0}) + b \frac{\partial h}{\partial \mathbf{x}}(\mathbf{0}) \right) \mathrm{d}\mathbf{x}(t_k). \tag{6.24}$$

Multiplying both sides of Eq. (6.23) by $\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})$, the following is obtained:

$$\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\, \mathrm{d}\mathbf{x}(t_{k+1}) = \left(\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\frac{\partial \Phi}{\partial \mathbf{x}}(\mathbf{0})\right)\mathrm{d}\mathbf{x}(t_k). \tag{6.25}$$

Consequently, from Eqs. (6.24) and (6.25), it can be deduced that the nonlinear transformation centered at the equilibrium point must satisfy the following (phase) condition:

$$\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\frac{\partial \Phi}{\partial \mathbf{x}}(\mathbf{0}) = A\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0}) + b\frac{\partial h}{\partial \mathbf{x}}(\mathbf{0}). \tag{6.26}$$

Notice that the elements of the Jacobian matrix $\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})$ can be computed by solving the system of equations (6.26), provided that Φ and h are explicitly available.

Physics-Informed Neural Network formulation

Physics-Informed Neural Networks (PINNs), introduced in [52, 184], provide a framework for incorporating known physical laws directly into the training of neural networks. By embedding governing equations or structural constraints into the loss function, PINNs enable the construction of models that remain consistent with the underlying system dynamics even when data are incomplete or noisy. A detailed overview of feedforward neural networks, their training procedures, and the PINN methodology is provided in Appendix E.

PINNs are here employed as a computational tool for approximating nonlinear transformation operators arising from the functional equations governing the observer design. In this setting, the unknown transformation is represented by a neural network whose parameters, collectively denoted by θ, are learned by minimizing a loss function that penalizes violations of the functional relations defining the transformation. This formulation enables the computation of the required transformation without relying on analytical derivations, while preserving the structural constraints imposed by the underlying system dynamics.

Neural approximation of the transformation operator

In this thesis work, the objective is not the solution of a partial differential equation but the computation of the nonlinear transformation operator $T(\mathbf{x})$ that satisfies the functional equations (6.12). To this end, a feedforward neural network is employed to approximate $T(\mathbf{x})$. The neural approximation of the transformation operator is denoted by $\hat{T}(\mathbf{x})$ and is constructed as:

$$\hat{T}(\mathbf{x}) = {W^{(0)}}^T S^{(2)}\left({W^{(2)}}^T S^{(1)}\left({W^{(1)}}^T x + \beta^{(1)}\right) + \beta^{(2)}\right) + \beta^{(0)}, \quad \hat{T} : \mathbb{R}^d \longrightarrow \mathbb{R}^d. \tag{6.27}$$

Specifically:

$W^{(1)} \in \mathbb{R}^{d\times N_1}$, $W^{(2)} \in \mathbb{R}^{N_1\times N_2}$, $W^{(0)} \in \mathbb{R}^{N_2\times d}$, where N_1 and N_2 denote the number of neurons in the first and second hidden layers, respectively. The vectors $\beta^{(1)} \in \mathbb{R}^{N_1}$, $\beta^{(2)} \in \mathbb{R}^{N_2}$, and $\beta^{(0)} \in \mathbb{R}^{d}$ are the corresponding bias terms. The activation functions associated with the hidden layers are denoted by $S^{(1)}(\cdot)$ and $S^{(2)}(\cdot)$ and are applied component-wise. The set of all trainable parameters of the neural network is collectively denoted by θ.

Physics-based loss construction

To compute the transformation operator $T(\mathbf{x})$ using the neural approximation $\hat{T}(\mathbf{x})$, the loss function is constructed so as to enforce:

(i) satisfaction of the functional Eqs. (6.12),

(ii) preservation of the equilibrium condition $T(\mathbf{0}) = \mathbf{0}$,

(iii) matching of the Jacobian at the equilibrium.

Let $\{\mathbf{x}_i\}_{i=1}^{M}$ denote a set of collocation points in the domain of interest. The residual associated with the functional Eq. (6.12) is defined componentwise as:

$$r_{ij}^{(1)}(\mathbf{x}_i, \hat{T}(\mathbf{x}_i)) = \hat{T}_j(\Phi(\mathbf{x}_i)) - \left(\alpha_j{}^T \hat{T}(\mathbf{x}_i) + b_j(h(\mathbf{x}_i))\right), \quad i = 1,\dots,M, \quad j = 1,\dots,d, \tag{6.28}$$

where α_j^T denotes the j-th row of matrix A and $b_j(\cdot)$ denotes the j-th component of the function $b(\cdot)$. The residual enforcing the equilibrium constraint $T(\mathbf{0}) = \mathbf{0}$ is:

$$r_j^{(2)}(\hat{T}_j(\mathbf{0})) = \hat{T}_j(\mathbf{0}), \quad j = 1,\dots,d. \tag{6.29}$$

In order to satisfy the linearization constraint Eq. (6.26), the Jacobian of the neural approximation at the equilibrium must match the solution of the phase condition. The corresponding residual is defined as:

$$r_{jq}^{(3)} = \frac{\partial \hat{T}_j}{\partial x_q}(\mathbf{0}) - \frac{\partial T_j}{\partial x_q}(\mathbf{0}), \quad j = 1,\dots,d, \quad q = 1,\dots,d. \tag{6.30}$$

The overall loss function is defined as:

$$\mathcal{L}(\theta) = \sum_{i=1}^{M}\sum_{j=1}^{d}\left(r_{ij}^{(1)}\right)^2 + \sum_{j=1}^{d}\left(r_j^{(2)}\right)^2 + \sum_{j=1}^{d}\sum_{q=1}^{d}\left(r_{jq}^{(3)}\right)^2. \tag{6.31}$$

The optimal set of neural network parameters θ is obtained by solving the minimization problem:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta). \tag{6.32}$$

Greedy Continuation Strategy

In practice, direct training of the neural network over a large domain may lead to poor convergence or entrapment in local minima, especially when the nonlinear transformation operator exhibits steep gradients or highly nonlinear behavior.

To address this issue, a greedy continuation strategy is adopted. Specifically, a sequence of nested domains is defined:

$$D_1 \subset D_2 \subset \cdots \subset D_K = D, \tag{6.33}$$

where D denotes the final domain of interest.

The training procedure proceeds sequentially:

- First, the neural network is trained over the smallest domain D_1.
- The optimized parameters obtained from D_1 are then used as initialization for training over the enlarged domain D_2.
- This process is repeated iteratively until the full domain D is covered.

This continuation strategy improves numerical stability and enhances convergence of the optimization procedure.

6.3.2 Inverse of the nonlinear transformation

Once the nonlinear transformation operator $T(\mathbf{x})$ has been computed, the observer expressed in the original coordinates Eqs. (6.16) requires evaluation of the inverse mapping $T^{-1}(\mathbf{z})$. To compute the inverse transformation numerically, consider the nonlinear equation:

$$G(\mathbf{x}) = T(\mathbf{x}(t_k)) - \mathbf{z}(t_k) = \mathbf{0}, \tag{6.34}$$

where $\mathbf{z}(t_k)$ is known and $\mathbf{x}(t_k)$ is to be determined. The solution of (6.34) is obtained via Newton's method:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) - \left(\frac{\partial G}{\partial \mathbf{x}}(\mathbf{x}(t_k))\right)^{-1} G(\mathbf{x}(t_k)). \tag{6.35}$$

The iterative procedure is continued until the residual norm falls below a prescribed tolerance (typically 10^{-6}). The convergence of the Newton iteration is ensured locally due to the local invertibility of the transformation operator $T(\mathbf{x})$ established in Theorem 6.2.1. The numerical framework developed in this section provides a general procedure for computing the nonlinear transformation operator $T(\mathbf{x})$ associated with exact observer linearization.

Before applying this methodology to the data-driven modeling framework introduced in Chapter 5, it is essential to assess its accuracy, robustness, and convergence properties in

controlled environments where analytic solutions are available. To this end, the proposed Taylor-based and PINN-based approaches are first validated on illustrative benchmark nonlinear systems. These benchmark problems serve to evaluate the correctness of the computed transformation operator, the stability properties of the resulting observer, and the numerical performance of the proposed solution strategy. The methodology is subsequently applied to the crowd dynamics problem.

6.4 Illustrative benchmark problems

As previously noted, the proposed scheme is first validated using two benchmark problems where the right-hand side of the system equations, the analytical solutions for the system of functional equations, and their inverses are all known.

6.4.1 Numerical setup

Moreover, in accordance with the theoretical framework outlined in reference [183], we also used the power series solution method delineated in section 6.3.1, to find the nonlinear transformation. In particular, we used sixth-order multivariate Taylor polynomial approximations. The method produces a system of linear algebraic equations involving the unknown Taylor coefficients of the nonlinear transformation $T(x_1, x_2)$. The solution to this system can be computed using MATLAB's symbolic toolbox, as explained in Section 6.3.1. We opt for a sixth-order multivariate Taylor polynomial expansion for two reasons: firstly, the number of coefficients aligns well with the number of unknowns in our PINN, facilitating a fair comparison between the two schemes. Secondly, solving the system for an expansion of this order is already computationally challenging, highlighting the underlying issues of this traditional methodological approach usually employed to solve this type of problems for high order series expansions.

6.4.2 PINN-based solution

We have used FNNs with two hidden layers, with each layer containing five neurons. We have used sigmoid activation functions $\sigma(\mathbf{x})$. In the training process, we computed the derivatives for the optimization process using finite differences for the LM algorithm and we have set the maximum function evaluations to $500,000$ while the maximum iteration parameter was set to $50,000$. the finite difference step was set to $1e^{-05}$ and the step size tolerance was set to $1e^{-12}$.

This realization prompted the exploration of alternative strategies to enhance the training process. Subsequently, the adoption of a greedy-wise procedure emerged as a pivotal step in refining the PINN's convergence behavior[125]. By incorporating this approach, the greedy-wise procedure effectively addressed challenges related to convergence of the PINN close to the singular points.

We initiated training within the region $[-0.1, 0] \times [-0.1, 0]$. Subsequently, we sys-

tematically expanded the grid size in the domain, with a varying step size in the grid depending on the specific benchmark problem at hand. In the first benchmark problem, we employed a grid step size of -0.1 in both directions, until reaching a domain size of $[-0.4, 0] \times [-0.4, 0]$, then the step size is decreased to a value of -0.01 until reaching $[-0.49, 0] \times [-0.49, 0]$. Finally, the step size is decreased again to a value of -0.001 until reaching the domain of interest, i.e. $[-0.495, 0] \times [-0.495, 0]$, while for the second benchmark problem, we initiated training within the region $[-0.1, 0] \times [-0.1, 0]$ and we adopted a step of -0.1 until reaching a grid of $[-0.8, 0] \times [-0.8, 0]$. After that we selected a step size of -0.01 until reaching the domain of interest $[-0.91, 0] \times [-0.91, 0]$. In both instances, following the completion of training within each subdomain, we utilized the learnable parameters specifically, the weights and biases obtained from the preceding subdomain as the initial guess for the optimization algorithm in the subsequent subdomain.

6.4.3 Benchmark problem 1

System definition and analytical transformation

The following nonlinear discrete-time system is considered with state-space representation:

$$\begin{aligned} x_1(t_{k+1}) &= \exp\left(0.2\frac{x_2(t_k)}{1+x_2(t_k)}\right)\sqrt[2]{1+x_1(t_k)+x_2(t_k)} - 1 - 0.4x_2(t_k) \\ &\quad -0.5\ln\left(1+x_1(t_k)+x_2(t_k)\right) \\ x_2(t_{k+1}) &= 0.5\ln\left(1+x_1(t_k)+x_2(t_k)\right) + 0.4x_2(t_k) \\ y(t_k) &= x_2(t_k), \end{aligned} \tag{6.36}$$

where the measured output variable y coincides with the second state variable x_2. For this benchmark problem, the system dimension is $d = 2$ and the measured output dimension is $p = 1$. Notice that $(x_1, x_2) = (0, 0)$ is an equilibrium point of Eq. (6.36) and $1 + x_1 + x_2 > 0$ for well-defined right-hand side functions. The Jacobian matrix F of Eq. (6.36) evaluated at the equilibrium is:

$$F = \begin{bmatrix} 0.0 & -0.2 \\ 0.5 & 0.9 \end{bmatrix}. \tag{6.37}$$

The eigenvalues of F are $f = 0.1298$, $f_2 = 0.7702$. Consider now the following choice for matrix A:

$$A = \begin{bmatrix} 0.5 & 0.3 \\ 0.5 & 0.4 \end{bmatrix}, \tag{6.38}$$

with eigenvalues:

$$\mu_1 = 0.8405, \qquad \mu_2 = 0.0515. \tag{6.39}$$

Moreover, the following output injection map was chosen:

$$b(y) = b(x_2) = \begin{bmatrix} 0.2\left(\frac{x_2}{1+x_2}\right) - 0.3x_2 \\ 0.0 \end{bmatrix}, \tag{6.40}$$

such that the pair (A, B) is controllable. Under the above choice for A, all other assumptions of Theorem 6.2.1 are satisfied. Consequently, the associated system of functional equations (6.12) takes the form:

$$\begin{aligned} T_1(u_1, u_2) &= 0.5T_1 + 0.3T_2 + 0.2\left(\frac{x_2}{1+x_2}\right) - 0.3x_2 \\ T_2(u_1, u_2) &= 0.5T_1 + 0.4T_2 \\ T_1(0,0) &= 0 \\ T_2(0,0) &= 0, \end{aligned} \tag{6.41}$$

where

$$u_1 = \exp\left(0.2\frac{x_2}{1+x_2}\right)\sqrt{1+x_1+x_2} - 1 - 0.4x_2 - 0.5\ln(1+x_1+x_2), \tag{6.42}$$

$$u_2 = 0.5\ln(1+x_1+x_2) + 0.4x_2. \tag{6.43}$$

Equation (6.12) admits a unique real analytic and locally invertible solution. In particular, the closed-form solution is:

$$T_1(x_1, x_2) = \ln(1+x_1+x_2), \qquad T_2(x_1, x_2) = x_2. \tag{6.44}$$

Since

$$\frac{\partial T}{\partial x}(0,0) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{6.45}$$

and

$$\det\left(\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\right) \neq 0, \tag{6.46}$$

the above transformation $T(\mathbf{x})$ is locally invertible around $\mathbf{x} = \mathbf{0}$.

According to Theorem 6.2.1, the observer in transformed variables is:

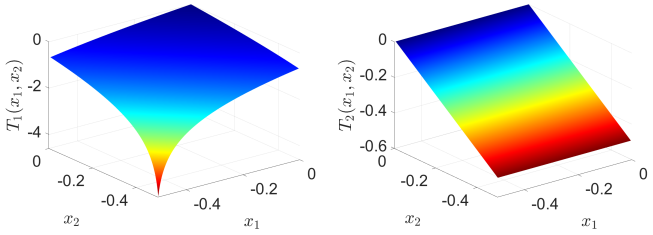
$$\begin{aligned} \hat{z}_1(t_{k+1}) &= 0.5\hat{z}_1(t_k) + 0.3\hat{z}_2(t_k) + 0.2\frac{y(t_k)}{1+y(t_k)} - 0.3y(t_k) \\ \hat{z}_2(t_{k+1}) &= 0.5\hat{z}_1(t_k) + 0.4\hat{z}_2(t_k). \end{aligned} \tag{6.47}$$

Through the inverse transformation, the state estimates in original variables are:

$$\bar{x}_1(t_k) = \exp(\hat{z}_1(t_k)) - \hat{z}_2(t_k) - 1$$

$$\bar{x}_2(t_k) = \hat{z}_2(t_k). \tag{6.48}$$

please notice that the nonlinear operator in the first benchmark example exhibits singularities when $x_1 + x_2 = -1$ in the system's state space. The primary objective is to learn an approximation of the transformation operator in the domain $[x_L, 0] \times [x_L, 0] = [-0.495, 0] \times [-0.495, 0]$. Figure (6.1) depicts the analytical solutions $T_1(x_1, x_2), T_2(x_1, x_2)$, of the associated system of functional Eqs. (6.41). We note that it encompasses the solution



(a) (b)

Figure 6.1: Analytical solution of the NFEs (6.41) in $[-0.495, 0] \times [-0.495, 0]$. (a) $T_1(x_1, x_2) = \ln(1 + x_1 + x_2)$. A steep-gradient at $(-0.495, -0.495)$ is due to the presence of a singular point at $(x_1, x_2) = (-0.5, -0.5)$. (b) $T_2(x_1, x_2) = x_2$.

.

domain which has been deliberately chosen to be $x_1, x_2 \in [-0.495, 0]$ since $T_1(x_1, x_2)$ exhibits a singular point at $(x_1, x_2) = (-0.5, -0.5)$. Therefore, the first reasonable step would be to comparatively evaluate the numerical approximation accuracy of the proposed PINN scheme in this region against the existing analytical solution and assess its impact on the performance profile of the resulting discrete-time observer.

Numerical results

Table (6.1) presents a comparison of L_1, L_2, L_∞ error norms between the analytical and numerical solutions on the test set. Specifically, it summarizes the results obtained with the 6th order power series expansions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$, and also with PINN solution approximations both trained in the entire domain at once using the greedy approach. The reported errors include the median, 5th percentile, and 95th percentile over 100 runs.

Complementing this comparison, table (6.2), on the other hand, reports the results of uncertainty quantification carried out to investigate the convergence of the PINN scheme in approximating the nonlinear transformation operator $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ on a test set involving the use of a grid of 20×20 Chebyshev-Lobatto distributed points. The investigation encompassed 100, 200, 300, and 400 independent runs, providing a thorough evaluation of the resulting solution's variability.

Furthermore, Figure (6.3) depicts the error characteristics between the analytical solution and the outcome generated by the PINN for $T_1(x_1, x_2)$. In this benchmark problem, the transformation operator is quite important as it encompasses the singular point and relates to the state assumed to be unmeasurable. In Figure (6.3)(a), we juxtapose the solution approximations obtained through training across the entire domain and in a greedy-wise manner. The L_1 norm for both PINN approximations is presented; the blue line signifies the median of the L_1 error evaluated at specific points in the domain. Additionally, the green band represents a confidence interval spanning from the 5th to the 95th percentile, reflecting a PINN trained in a greedy-wise fashion. On the other hand, the black line depicts the median of the L_1 error when the network is trained across the entire domain at once, and the green band denotes a confidence interval ranging from the 5th to the 95th percentile.

Figure (6.3)(b), depicts the L_2 norm; the blue line depicts the median of the L_2 error with a confidence interval ranging from the 5th to the 95th percentile represented by the green band; the black line shows again the median of the L_2 error when the network is trained across the entire domain at once, considering the blue band as a confidence interval in the same sense as before. Figure (6.3)(c) displays results for the L_∞ norm. It can be inferred that the greedy-wise training significantly outperforms the one obtained when training the network across the entire domain in a single step. The numerical approximation from the single training exhibits a poorer performance, particularly in regions proximate to the singular point, as is also evident in Figure (6.2).

Table 6.1: Benchmark problem 1. Test set using 20×20 Chebyshev-Lobatto distributed points L_1, L_2, L_∞ error norms (median, 5th- 95th percentiles) between the analytical and computed solutions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ of the various schemes trained in the entire domain and with the greedy approach, over 100 runs.

$T(x_1, x_2)$	Error norm	Power-Series $6th$ order -	PINN Entire domain Median (5th,95th)	PINN Greedy Median (5th,95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	6.89E+01	1.186E+01,(1.048E+01,1.372E+01)	6.051E−01,(6.94E−02,6.00E+00)
	$\|\cdot\|_2$	4.55E+00	1.062E+01,(9.94E+00,1.243E+01)	1.528E−01,(1.62E−02,1.33E+00)
	$\|\cdot\|_\infty$	2.88E+00	8.28E+00,(7.00E+00,9.63E+00)	9.15E−02,(7.80E−03,5.87E−01)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	0	2.44E+00,(7.40E−01,4.00E+00)	3.595E−01,(4.39E−02,2.90E+00)
	$\|\cdot\|_2$	0	2.34E+00,(6.68E−01,3.75E+00)	6.31E−02,(1.02E−02,4.988E−01)
	$\|\cdot\|_\infty$	0	1.65E+00,(4.55E−01,2.68E+00)	2.94E−02,(3.90E−03,1.85E−01)

Finally, Figure (6.3)(d) depicts the difference between the analytical inverse operator $T_1^{-1}(x_1, x_2)$ (6.57) of the observed variable (6.56) and the numerical approximation of the inverse $\hat{T}_1^{-1}(x_1, x_2)$ as computed by Newton's method (presented in Section 6.3.2). As shown, the difference between the true state and the inverse transformation value generated by the PINN converges to zero. For this particular simulation, we have initialized the z-states as follows: $z_1(1) = 0$ and $z_2(1) = 0$, and we have used as initial guess for Newton's method the following values for the x-states: $\mathbf{x_0} = [0.1, 0.1]$. Moreover, we have initialized the analytical inverse expression using: $\hat{x}_1(1) = 0$ and $\hat{x}_2(1) = 0$; the actual states were: $x_1 = -0.495$ and $x_2 = 0.35$. Regarding the Newton's method, we

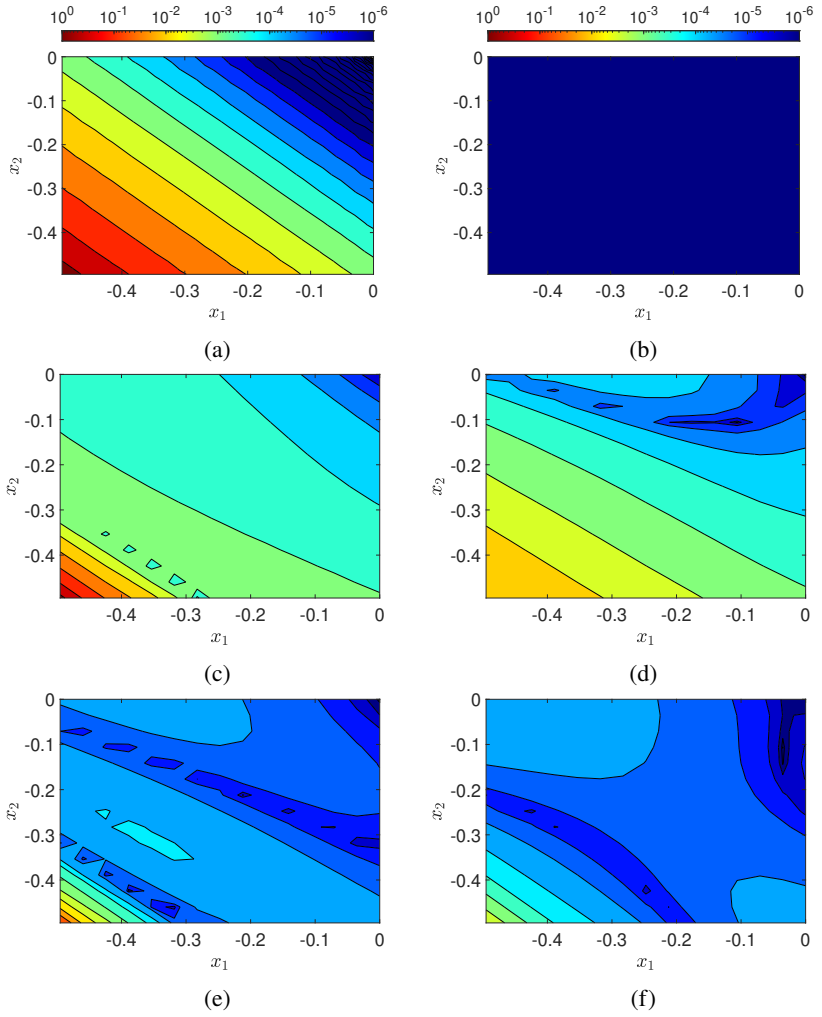


Figure 6.2: Benchmark problem 1. Test sets (grids of 20×20 Chebyshev-distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the Eq. (6.36) in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PINN trained in the entire domain $[-0.495, 0] \times [-0.495, 0]$. (e),(f) PINN trained via the greedy-wise approach.

have set the relative and absolute tolerances at: $tol < 1E\text{-}06$ (the scheme converged in 5 iterations). The numerical results for the train set are depicted in Appendix F.

Table 6.2: Benchmark problem 1. Uncertainty quantification systematically conducted on a test set employing a grid of 20×20 Chebyshev-Lobatto distributed points to investigate the convergence behavior of the Physics-Informed Neural Networks (PINN). The study involves performing 100, 200, 300, and 400 independent runs to thoroughly assess the variability in the results. The analysis focused on capturing the central tendency by calculating the median and examining the dispersion of outcomes through the 5th and 95th percentile confidence intervals.

$T(x_1, x_2)$	Error norm	100 runs Median (5th,95th)	200 runs Median (5th,95th)	300 runs Median (5th,95th)	400 runs Median (5th,95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	6.051E−01, (6.94E−02,6.00E+00)	6.34E−01, (6.20E−02,5.60E+00)	6.27E−01, (6.188E−02,5.45E+00)	6.12E−01, (6.12E−02,5.32E+00)
	$\|\cdot\|_2$	1.528E−01, (1.62E−02,1.33E+00)	1.77E−01, (1.95E−02,1.23E+00)	1.72E−01, (1.88E−02,1.21E+00)	1.705E−01, (1.76E−02,1.18E+00)
	$\|\cdot\|_\infty$	9.15E−02, (7.80E−03,5.87E−01)	8.87E−02, (6.58E−03,5.12E−01)	8.75E−02, (6.55E−03,5.10E−01)	8.66E−02, (6.502E−03,5.108E−01)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	3.595E−01, (4.39E−02,2.90E+00)	3.90E−01, (4.98E−02,3.25E+00)	3.84E−01, (4.85E−02,3.21E+00)	3.78E−01, (4.78E−02,3.16E+00)
	$\|\cdot\|_2$	6.31E−02, (1.02E−02,4.988E−01)	6.21E−02, (1.80E−02,6.01E−01)	6.19E−02, (1.75E−02,6.00E−01)	6.15E−02, (1.69E−02,5.98E−01)
	$\|\cdot\|_\infty$	2.94E−02, (3.90E−03,1.85E−01)	2.81E−02, (3.28E−03,2.10E−01)	2.75E−02, (3.22E−03,2.03E−01)	2.701E−02, (3.19E−03,1,98E−01)

6.4.4 Benchmark problem 2

System definition and analytical transformation

Consider also the following nonlinear discrete-time dynamical system [185]:

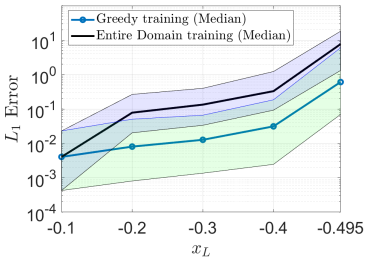
$$\begin{aligned} x_1(t_{k+1}) &= \frac{0.5\frac{x_1(t_k)}{1+x_1(t_k)} - 0.9x_2(t_k)}{1 - 0.5\frac{x_1(t_k)}{1+x_1(t_k)} + 0.9x_2(t_k)} \\ x_2(t_{k+1}) &= x_2(t_k) \\ y(t_k) &= x_1(t_k) \end{aligned} \tag{6.49}$$

where x_2 represents an unidentified constant parameter or disturbance term which needs to be estimated and x_1 the measured state variable. Please notice that $(x_1, x_2) = (0, 0)$ is an equilibrium point of the above system and the Jacobian matrix F evaluated at this point is:

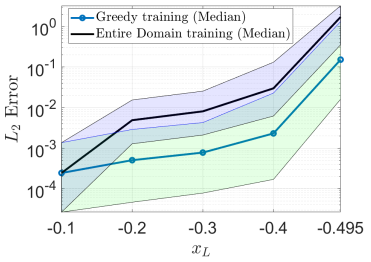
$$F = \begin{bmatrix} 0.5 & -0.9 \\ 0.0 & 1 \end{bmatrix}. \tag{6.50}$$

The eigenvalues of F are: $f_1 = 0.5$ and $f_2 = 1$, and therefore, the original nonlinear discrete-time observer design method presented in [183] can not be applied since the second eigenvalue is not located within the Poincaré domain; instead it lies on the unit circle. However, under the following choice for the matrix A

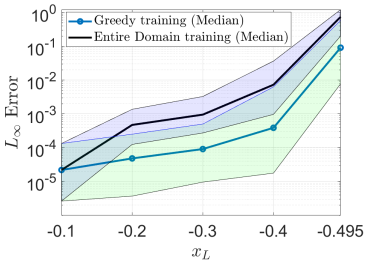
$$A = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \tag{6.51}$$



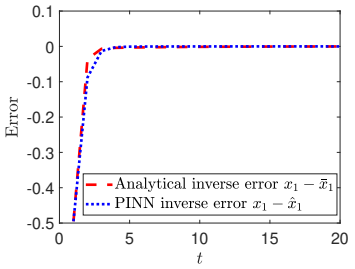
(a) Greedy vs Single training, L_1 norm



(b) Greedy vs Single training, L_2 norm



(c) Greedy vs Single training, L_∞ norm



(d) Analytical vs PINN-based inverse error

Figure 6.3: Benchmark Problem 1: (a)-(c) Panels involve a comprehensive comparison between the performance of Physics-Informed Neural Networks (PINN) trained using two distinct approaches: one trained on the entire domain at once (Black line) and another trained using a greedy approach (blue line). The evaluation is conducted on a test set comprising 20×20 Chebyshev nodes of the second type. The reported performance metrics include the L_1, L_2, and L_∞ error norms, presented in terms of the median and the 5th to 95th percentiles. The evaluation is performed over 100 independent runs for the transformation $T_1(x_1, x_2)$. Panel (d) depicts a comparison between observation errors. The red line represents the difference between the analytical inverse $\bar{x}_1$ of the transformation $T_1(x_1, x_2)$ and the real state x_1, whereas the blue line depicts the difference between the true state x_1 and its approximation obtained through Newton's method, denoted as $\hat{x}_1$.

with eigenvalues: $\mu_1 = 0$ and $\mu_2 = 0.1$, and an output injection map:

$$b(y) = b(x_1) = \begin{bmatrix} 0.5\left(\frac{x_1}{1+x_1}\right) \\ \left(\frac{x_1}{1+x_1}\right) \end{bmatrix} \tag{6.52}$$

all assumptions in the method's extension presented in [185] (where the Poincaré domain assumption was relaxed and replaced by a more general one of the Siegel-type) are met, and therefore, the proposed nonlinear discrete-time observer exists and can be similarly

designed. Indeed, the associated system of functional equations takes the following form:

$$\begin{aligned} T_1\left(\frac{0.5\frac{x_1}{1+x_1}-0.9x_2}{1-0.5\frac{x_1}{1+x_1}+0.9x_2},x_2\right) &= 0.5\frac{x_1}{1+x_1} \\ T_2\left(\frac{0.5\frac{x_1}{1+x_1}-0.9x_2}{1-0.5\frac{x_1}{1+x_1}+0.9x_2},x_2\right) &= 0.1T_2+\frac{x_1}{1+x_1} \\ T_1(0,0) &= 0 \\ T_2(0,0) &= 0, \end{aligned} \tag{6.53}$$

The above system of functional Eqs. (6.53) admits a unique analytically derived closed-form solution shown below:

$$\begin{aligned} T_1(x_1,x_2) &= \frac{x_1}{1+x_1}+0.9x_2 \\ T_2(x_1,x_2) &= \frac{5}{2}\left(\frac{x_1}{1+x_1}+x_2\right). \end{aligned} \tag{6.54}$$

Since:

$$\frac{\partial T}{\partial \mathbf{x}}(0,0)=\begin{bmatrix}1 & 0.9\\ 2.5 & 2.5\end{bmatrix}, \tag{6.55}$$

and $det[T] \neq 0$, the above solution $T(\mathbf{x})$ is locally invertible around the origin. The proposed discrete-time observer is then given by the following dynamic equations:

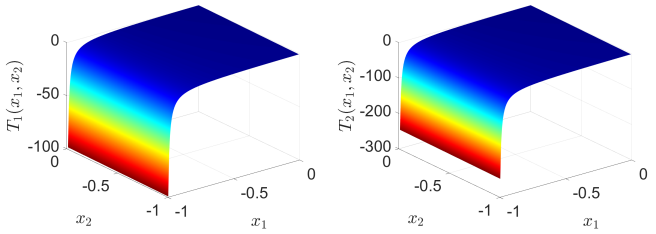
$$\begin{aligned} \hat{z}_1(t_{k+1}) &= 0.5\frac{y(t_k)}{1+y(t_k)} \\ \hat{z}_2(t_{k+1}) &= 0.1\hat{z}_2(t)+\frac{y(t_k)}{1+y(t_k)}. \end{aligned} \tag{6.56}$$

Therefore, using the inverse transformation map, the state estimates expressed in the original state variables are given by the following equations:

$$\begin{aligned} \bar{x}_1(t_k) &= \frac{10\hat{z}_1(t_k)-3.6\hat{z}_2(t_k)}{1-10\hat{z}_1(t_k)+3.6\hat{z}_2(t_k)} \\ \bar{x}_2(t_k) &= 4\hat{z}_2(t_k)-10\hat{z}_1(t_k). \end{aligned} \tag{6.57}$$

Numerical results

In this second benchmark problem, the nonlinear transformation operator exhibits a singularity when $x_1 = -1$. Here, we aim to learn this map in the domain $[x_L, 0]\times[x_L, 0] = [-0.91, 0]\times[-0.91, 0]$. Figure (6.4) depicts the analytical solutions $T_1(x_1, x_2)$, $T_2(x_1, x_2)$, of the associated system of functional Eqs. (6.53). Please notice that it includes the



(a) (b)

Figure 6.4: Analytical solution of the functional equations (6.41). (a) $T_1(x_1, x_2) = \frac{x_1(t)}{1+x_1(t_k)} + 0.9x_2$. (b) $T_2(x_1, x_2) = \frac{5}{2}\left(\frac{x_1(t_k)}{1+x_1(t_k)} + x_2\right)$. A steep-gradient at $(-0.91, -0.91)$ is due to the presence of a singular point at $x_1 = -1$.

solution domain, which in this case is (x_1, x_2), $\in [-0.91, 0]$ because $T_1(x_1, x_2)$ displays a singular point at $(x_1, x_2) = (-1, -1)$. As in the previous benchmark example, the focus is placed on the evaluation of the impact of the proposed PINN scheme on the performance profile of the resulting nonlinear observer by comparing the identified transformation operator to the analytical one and hence, demonstrating the enhanced numerical approximation accuracy in this domain attained by PINN.

We utilized the Matlab deep-learning toolbox, with the aid of which a custom code was developed to implement the Physics-Informed Neural Network (PINN) strategy. In this process, we employed the Levenberg-Marquardt (LM) algorithm to optimize the learnable parameters. The scheme utilizes finite differences for computing the required derivatives. However, it's worth noting that these derivatives can also be computed analytically, numerically or through Automatic Differentiation (AD), as demonstrated in [186]. The simulation results obtained when the PINN was trained in the entire domain without the greedy approach are also presented in order to evaluate the efficacy of the proposed greedy strategy. For training, we used 15×15 equispaced distributed points in the interval $[a, b]$: $a = -0.495$, $b = 0$, for the model (6.36), and $a = -0.91$, $b = 0$, for the model (6.49). For the test set, we employed Chebyshev-Lobatto nodes, specifically selecting the roots of the (second kind) Chebyshev-Lobatto polynomial of degree h, i.e.

$$x_n = \frac{1}{2}(a + b) - \frac{1}{2}(a - b)cos\left(\frac{2n - 1}{2n}\pi\right) \quad n = 1, \cdots, h \tag{6.58}$$

This particular benchmark problem introduces the challenge of unavailability of the state

x_2, with only the state x_1 being measurable. Consequently, the focus of this problem is on the reconstruction of the second state. As highlighted earlier, the unmeasured state x_2 is viewed as either an unidentified system parameter or unknown disturbance, thus adding complexity to the estimation task. Table (6.4), illustrates the results of uncertainty quantification performed to investigate the convergence of the PINN scheme in approximating the nonlinear transformation maps $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ on a test set (a grid of 20×20 Chebyshev-Lobatto distributed points). In our computations, we considered 100, 200, 300, and 400 MC runs. As shown, the medians and variances, are for any practical means of the same order, thus indicating convergence of the training process.

Figure (6.6) depicts the approximation error behavior between the analytical solution and the one produced by PINN for $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$. Please notice that $T_2(x_1, x_2)$ is the transformation map of interest in this problem because it is related to the state we seek to observe as mentioned earlier. The approximation accuracy of the PINN trained both across the entire domain and in a greedy-wise manner is assessed. Figure (6.6)(a) shows the L_1 norm of the transformation map $T_1(x_1, x_2)$. The blue line signifies the median, and the green band represents a confidence interval spanning the 5th to the 95th percentile range for the PINN model trained in a greedy-wise fashion. The black line and the blue band corresponds to the PINN model trained across the entire domain without the greedy approach. Figure (6.6)(b) depicts the corresponding L_2 error norms, and Figure (6.6)(c) depicts the corresponding L_∞ error norms. Figure (6.6)(d-f) depict similar results for $T_2(x_1, x_2)$.

It is evident, that the greedy-wise training procedure outperforms the single training one for the approximation of $T_2(x_1, x_2)$. Finally, Figure (6.7) depicts the error between the trajectories computed by applying the analytically derived inverse $T^{-1}(x_1, x_2)$ (6.57) to the observer (6.56) and the numerical approximation of the inverse $\hat{T}^{-1}(x_1, x_2)$ computed by Newton's method. For this task, we have initialized the z-states as: $z_1(1) = 0$ and $z_2(1) = 0$, and we have used as an initial guess the values $\mathbf{x_0} = [0.1, 0.1]$. The analytical inverse expression was initialized using the values: $\hat{x}_1(1) = 1$ and $\hat{x}_2(1) = -0.4$. Moreover, as mentioned earlier, the tolerance level for Newton's method was set at tol <1E−06. The numerical results for the train set are depicted in Appendix F.

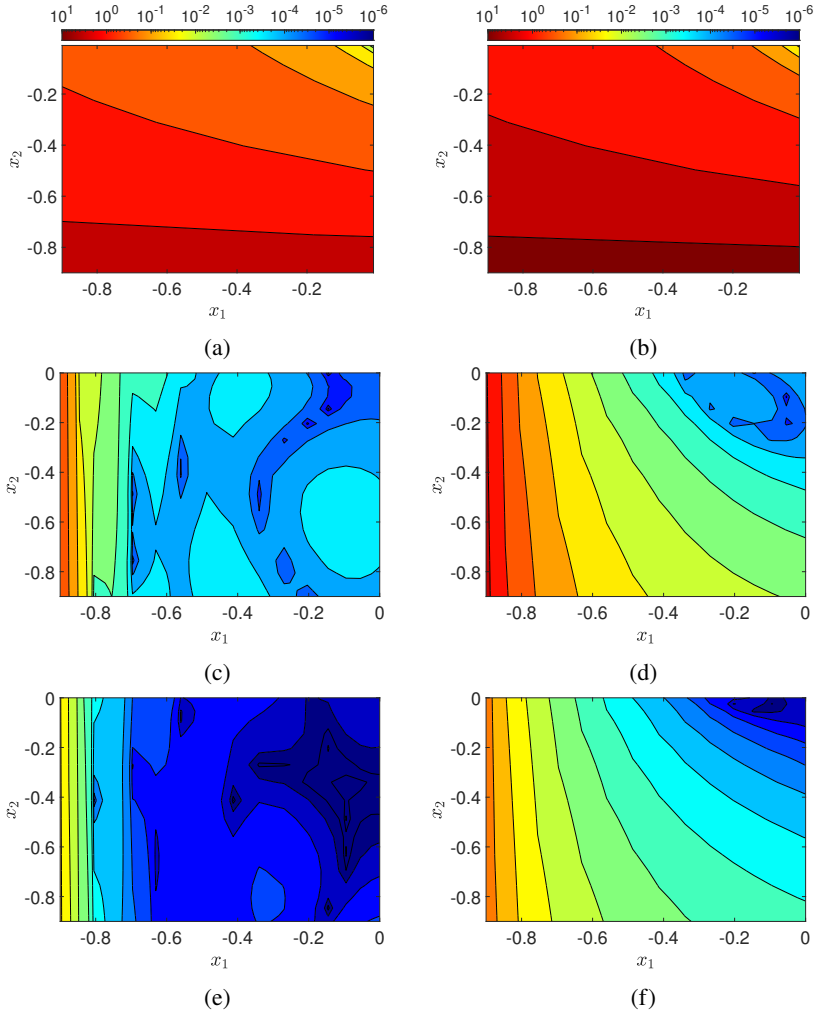


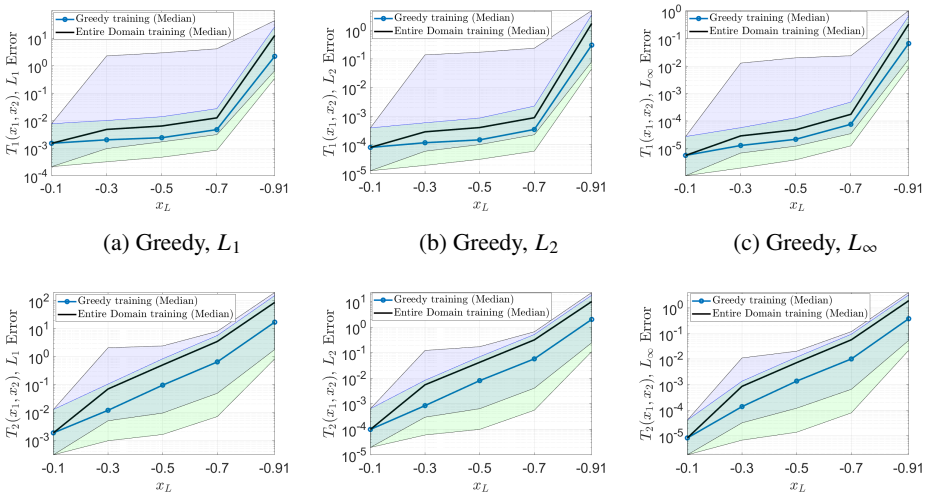
Figure 6.5: Test sets (grids of 20×20 Chebyshev-Lobatto distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the E Eq. (6.3) in $[-0.91, 0] \times [-0.91, 0]$. (c),(d) PINN trained over the entire domain $[-0.91, 0] \times [-0.91, 0]$. (e),(f) PINN trained via the greedy-wise approach.

Table 6.3: Test sets consist of grids with dimensions of 20×20 Chebyshev-distributed points. Error norms, including L_1, L_2, and L_∞, quantify the disparities between the analytical and computed solutions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ using various training schemes. These schemes encompass training both with a greedy-wise approach and across the entire domain $[-0.91, 0] \times [-0.91, 0]$. The analysis of error norms includes statistics such as the median and 95th percentiles, aggregated over 400 runs, for each norm corresponding to both $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$.

$T(x_1, x_2)$	Error norm	Power-Series $6th$ order	PINN Entire domain Median(5th,95th)	PINN Greedy Median(5th,95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	5.19E+02	7.03E+01,(5.75E+01, 8.31E+01)	2.17E+00,(3.43E−01, 2.60E+01)
	$\|\cdot\|_2$	4.13E+01	2.16E+01,(1.73E+01,2.62E+01)	3.55E−01,(4.52E−02,3.48E+00)
	$\|\cdot\|_\infty$	1.71E+01	9.39E+00,(7.32E+00,1.16E+01)	3.37E−01,(1.78E−02,1.058E+00)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	5.87E+01	2.14E+02,(1.82E+02, 2.15E+02)	1.72E+01,(8.25E−01, 7.50E+01)
	$\|\cdot\|_2$	5.30E+01	6.92E+01,(5.84E+01,8.04E+01)	2.02E+00,(1.11E−01,1.92E+01)
	$\|\cdot\|_\infty$	4.40E+01	3.40E+01,(2.89E+01,4.02E+01)	1.90E+00,(5.36E−02, 3.99E+00)

Table 6.4: Benchmark problem 2. Uncertainty quantification systematically conducted on a test set employing a grid of 20×20 Chebyshev-Lobatto distributed points to investigate the convergence behavior of the Physics-Informed Neural Networks (PINN). The study involves performing 100, 200, 300, and 400 independent runs to thoroughly assess the variability in the results. The analysis focused on capturing the central tendency by calculating the median and examining the dispersion of outcomes through the 5th and 95th percentile confidence intervals.

$T(x_1, x_2)$	Error norm	100 runs Median (5th,95th)	200 runs Median (5th,95th)	300 runs Median (5th,95th)	400 runs Median (5th,95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	2.17E+00, (3.43E−01,2.60E+01)	2.88E+00, (3.97E−01, 2.81E+01)	2.83E+00, (3.86E−01, 2.721E+01)	2.72E+00, (3.81E−01, 2.715E+01)
	$\|\cdot\|_2$	3.55E−01, (4.52E−02,3.48E+00)	4.23E−01, (5.12E−02,3.93E+00)	4.01E−01, (5.07E−02,3.66E+00)	3.98E−01, (4.93E−02,3.563E+00)
	$\|\cdot\|_\infty$	3.37E−01, (1.78E−02,1.0581E+00)	3.90E−01, (1.69E−02,1.210E+00)	3.84E−01, (1.03E−02,7.01E−01)	3.81E−01, (1.00E−02,6.91E−01)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	1.72E+01, (8.25E−01, 7.50E+01)	1.95E+01, (8.76E−01, 7.85E+01)	1.49E+01, (8.39E−01, 6.55E+01)	1.20E+01, (8.32E−01, 6.13E+01)
	$\|\cdot\|_2$	2.02E+00, (1.11E−01,1.92E+01)	2.32E+00, (1.52E−01,2.05E+01)	1.787E+00, (1.23E−01,1.786E+01)	1.59E+00, (1.09E−01,1.778E+01)
	$\|\cdot\|_\infty$	1.90E+00, (5.36E−02, 3.99E+00)	1.92E+00, (5.37E−02, 4.00E+00)	1.72E+00, (5.64E−02, 3.21E+00)	1.49E+00, (5.35E−02, 3.20E+00)



(a) Greedy, L_1 (b) Greedy, L_2 (c) Greedy, L_∞

(d) Single, L_1 (e) Single, L_2 (f) Single, L_∞

Figure 6.6: Benchmark problem 2. (a)-(c) Panels involve a comprehensive comparison between the performance of Physics-Informed Neural Networks (PINN) trained using two distinct approaches: one trained on the entire domain at once (Black line) and another trained using a greedy approach (blue line). The evaluation is conducted on a test set comprising 20×20 Chebyshev nodes of the second type. The reported performance metrics include the L_1, L_2, and L_∞ error norms, presented in terms of the median and the 5th to 95th percentiles. The evaluation is performed over 100 independent runs for the transformation $T_1(x_1, x_2)$. Furthermore Panels (d)-(f) depicts the same error norms with the same configuration for the transformation $T_2(x_1, x_2)$.

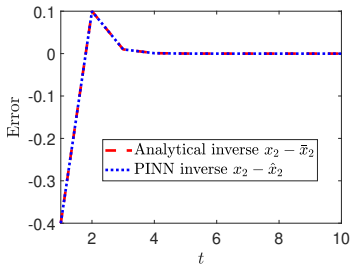


Figure 6.7: Comparison between observation errors. The red line represents the difference between the analytical inverse $\bar{x}_2$ of the transformation $T_2(x_1, x_2)$ and the real state x_2, whereas the blue line depicts the difference between the true state x_2 and its approximation obtained through Newton's method, denoted as $\hat{x}_2$.

Remark 6.4.1. These extensive numerical experiments highlight the importance of validating the proposed PIML framework in controlled settings where both the analytical dynamical system and the corresponding nonlinear functional equations and their solutions are known. Such benchmarks provide a rigorous basis for assessing the accuracy, stability, and reliability of the learned transformations before applying the methodology to systems whose dynamics are not available in closed analytical form.

Having established this validation through extensive numerical experiments and uncertainty quantification, the framework can now be integrated with the data-driven modeling pipeline developed in the previous chapters.

6.5 Integration with the data-driven multiscale modeling framework

We now return to the central application of this thesis wotk: the macroscopic modeling of crowd dynamics through reduced latent representations and learned evolution operators. In Chapter 4, we constructed a mass-preserving framework for representing density fields in a appropriate reduced latent space, while Chapter 5 introduced a data-driven discrete-time solution operator Φ_{ROM} governing the evolution of the reduced latent state.

Unfortunately in practical scenarios, as mentioned before, full access to the latent state vector is rarely available. Instead, only a subset of POD coefficients or equivalently partial information about the macroscopic density field may be accessible due to sensing or measurement constraints. The objective of this section is therefore to deploy the nonlinear observer framework developed in Sections 6.2–6.3 in order to reconstruct the complete latent state of the learned macroscopic crowd operator from partial observations.

Specifically, the trained LSTM-based reduced-order model is interpreted as a nonlinear discrete-time dynamical system evolving in the reduced POD space. Within this representation, we examine the structural properties required for nonlinear observer design, construct the associated transformation operator using the proposed PINN-based methodology, and evaluate the resulting observer in closed-loop simulations on unseen crowd trajectories.

Through this deployment, the learned macroscopic evolution operator developed in the previous chapters, culminating in Chapter 5, and the observer framework formulated in the present chapter are unified into a coherent methodology for state reconstruction in macroscopic crowd dynamics.

6.5.1 Reduced latent dynamics and observation structure

We briefly recall the first crowd case study (unidirectional flow) introduced in Chapter 5, where pedestrian trajectories generated by the SFM were mapped to macroscopic density fields and subsequently embedded into a reduced latent space using POD. Retaining 99% of the singular value energy resulted in a latent dimension $d = 6$, providing an efficient representation of the macroscopic density evolution.

Here, in contrast to chapter 5, a single-lag LSTM-based reduced-order model was trained, in this case, to approximate the discrete-time solution operator of the latent dynamics. This setup effectively constructs a discrete time-stepper that predicts the system state at the next time step based uniquely on the previous state. Thus, critical factor in this formulation is the choice of the macroscopic time step Δt used to subsample the density observations. Since the LSTM-based ROM relies only on a single lag of history, Δt must be selected to ensure that the stepper accurately captures the temporal evolution of the density field. Thus in our case we have selected $\Delta t = 0.35$. The resulting model can be written in the canonical nonlinear form Eq. 6.3. In this case, $\mathbf{x}(t_k) \in \mathbb{R}^d$ denotes the full vector of POD coefficients and $\mathbf{y}(t_k) \in \mathbb{R}^p$ represents the observable subset, with $p = 6 < d$. Since this is a multivariate problem, we introduce the observation matrix $H \in \mathbb{R}^{p\times d}$ that extracts the measured coefficients, that is. $\mathbf{y}(t_k) = H\mathbf{x}(t_k)$. As discussed in section 6.1, the origin $\mathbf{x} = \mathbf{0}$ corresponds to the uniform equilibrium density field and satisfies $\Phi(\mathbf{0}) = \mathbf{0}$, ensuring that the reduced latent dynamics admit a physically meaningful equilibrium suitable for local observer analysis.

6.5.2 Verification of observer assumptions

In order to apply the nonlinear observer framework introduced in Section 6.2, it is first necessary to verify that the learned latent dynamics satisfy the structural assumptions required for the existence of the transformation $T(\mathbf{x})$. In the present case, the right-hand side of the ROM is not analytically accessible, unlike in the previous examples. Therefore, the Jacobian matrix F defined in Eq. 6.5, corresponding to the learned evolution operator, is computed numerically at the equilibrium $\mathbf{x} = \mathbf{0}$ using a central finite-difference scheme with step size $h = 10^{-5}$.

The eigenvalues of F characterize the local behavior of the latent dynamics near the uniform density equilibrium. As illustrated in Fig. 6.8, these eigenvalues lie in the Poincaré domain, thereby satisfying Assumption 6.2.1. The observation matrix $H \in \mathbb{R}^{p\times d}$ is defined as a linear selection operator extracting the $d = 6$ measured latent variables. Each row of H contains a single nonzero entry equal to one, corresponding to the observable components of the latent state. Local observability around the equilibrium is assessed through the observability matrix Eq. (6.7)

For the present configuration, the observability matrix Eq. (6.7) has full rank d, ensuring that the pair (Φ, h) is locally observable around $\mathbf{x} = \mathbf{0}$ and satisfying Assumption 6.2.2. The input matrix B is simply chosen as $B = H^\top \in \mathbb{R}^{d\times p}$, so that correction terms act directly on the observable coordinates. The observer matrix $A \in \mathbb{R}^{d\times d}$ is constructed

to satisfy three requirements: (i) Schur stability, (ii) controllability of the pair (A, B), and (iii) spectral separation from the Jacobian F. Therefore in order to enforce Schur stability, a randomly initialized matrix is rescaled so that its spectral radius satisfies

$$\rho(A) = \gamma < 1, \tag{6.59}$$

with $\gamma = 0.8$ in the present study. In this case, the controllability matrix C has full rank d, as a result the system is controllable. Finally, spectral separation is enforced by requiring

$$\min_{i,j} |\lambda_i(A) - \mu_j(F)| \geq \delta, \tag{6.60}$$

with $\delta = 0.01$, ensuring the non-resonance condition necessary for the existence and uniqueness of the transformation operator, see Fig 6.8.

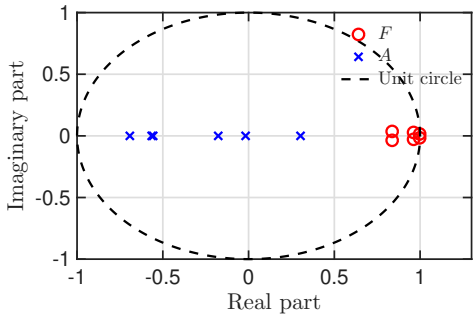


Figure 6.8: Spectral separation between eigenvalues of the Jacobian F (red circles) and the observer matrix A (blue crosses).

6.5.3 PINN-based approximation of the transformation $T(\mathbf{x})$

Having verified the structural assumptions required for nonlinear observer design, we now compute the approximation of the transformation operator $T(\mathbf{x})$ satisfying the functional relation in (6.12). Since the operator Φ is available only in numerical form via the data-driven framework introduces in this thesis work, the functional equations (6.12) cannot be solved analytically. We therefore employ the PINNs-based methodology introduced in Section 6.3 to approximate the transformation operator in the latent space.

The transformation $T : \mathbb{R}^d \to \mathbb{R}^d$ is approximated by a FNN with two hidden layers of 16 neurons each and hyperbolic tangent activation functions, followed by a linear output layer, using the network architecture and notation follow Eq. (6.27). The training domain is defined as a neighborhood of the equilibrium $\mathbf{x} = \mathbf{0}$ corresponding to the uniform density state. Collocation points are selected from the latent trajectories obtained from $C = 20$ different crowd simulations as in Chapter 5. For each training dataset, 60% of the K available time samples are selected without replacement, yielding a total of $\mathcal{T}_{tr} = 0.6\,K\,C$, where $C = 20$ trajectories and $k = 714$ timesteps as we have increased

the macroscopic time step $\Delta_t = 0.35$s. As a result, the solution is sought in 8568 training collocation points. For the testing data set we employ the other $\mathcal{T}_{ts} = 0.4\,K\,C$, collocation points.

In the training process, the derivatives required by the optimization were computed using finite differences within a custom `lsqnonlin`-based routine in MATLAB as during the benchmark problem this computational enviorment performed the best. The maximum number of iterations was set to 80, while the maximum number of function evaluations was set to 3×10^6. The function and step tolerances were both set to 10^{-8}. Forward finite differences were used with a step size of 10^{-4}. This value was selected empirically through trial and error, as it provided more stable derivative approximations during the optimization process.

The neural approximation $\hat{T}(\mathbf{x})$ is obtained by minimizing a slight variation of the loss function defined in Eq. (6.31). In this case, the true jacobian cannot be accessed because an analytical form of the macroscopic equation governing the pedestrian collective dynamics is not available. Consequently, the exact Jacobian $\nabla T(\mathbf{0})$ is unknown, and the corresponding residual term in the loss function cannot be computed directly.

$$r^{(3)}_{jl} = \frac{\partial \hat{T}_j}{\partial x_l}(0) - \left[\frac{\partial T_j}{\partial x_l}(0)\right], \qquad j,l = 1,\dots,d, \tag{6.61}$$

However, one can exploit the fact that the true Jacobian $J = \nabla T(0)$ satisfies the linearized functional equation

$$JF = AJ + BH. \tag{6.62}$$

This motivates replacing $r^{(3)}$ by the residual of this Sylvester-type condition evaluated at the network Jacobian $\hat{J} = \nabla \hat{T}(0)$:

$$r^{(3)}_{jl} = \left(\nabla \hat{T}(\mathbf{0})\,F - A\,\nabla \hat{T}(\mathbf{0}) - B\,H\right)_{jl}, \qquad j,l = 1,\dots,d. \tag{6.63}$$

In this way, the Jacobian of the approximated transformation $\hat{T}(\mathbf{x})$ by the PINN is constrained to satisfy the same structural relation as the true operator, even though $\nabla T(0)$ is unknown. Importantly, under a spectral separation condition between F and A enforced by Eq. (6.60), this Sylvester equation admits a unique solution for J, ensuring that the residual is both well-posed and consistent with the operator structure.

6.5.4 Numerical results

Due to the absence of true transformation $T(\mathbf{x})$, we first evaluate the functional residual on testing collocation points. Figure 6.9 reports the distribution of $\left\|r^{(1)}(\mathbf{x})\right\|_2$ across the testing dataset $\mathcal{X}_{ts}$. The residual norms remain uniformly small over unseen samples, indicating that the learned transformation satisfies the nonlinear functional relation with high accuracy in the dynamically relevant region of the latent space. At equilibrium, the pinning constraints are also satisfied with $\left\|r^{(2)}\right\|_2 = 5.53 \times 10^{-4}$ and $\left\|r^{(3)}\right\|_F = 8.7 \times$

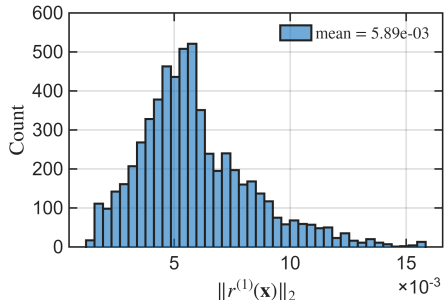


Figure 6.9: Validation of the learned transformation operator. Distribution of the functional residual $r^{(1)}(\mathbf{x})$ on testing collocation points, demonstrating enforcement of the nonlinear functional relation.

10^{-2}, confirming that the equilibrium invariance and Jacobian consistency conditions are effectively enforced.

To recover the latent state $\mathbf{x}(t_k)$ from the observer state $\mathbf{z}(t_k)$, the inverse of the learned transformation $\hat{T}(\mathbf{x})$ must be computed at each time step following the procedure in section 6.3.2 since $\hat{T}$ is not available in closed form. Finally, we deploy the complete observer framework to reconstruct the evolution of the pedestrian density field from partial measurements. The observer operates sequentially in time and receives only the available measurements of the observable latent variables $\mathbf{y}_k$, which in this case correspond to the first three latent states, i.e., $p = 3$. The observer is initialized from the first available measurement and then evolves according to the linear observer dynamics Eq. (6.15). At each time step, the estimate of the full latent state is obtained by solving the inverse transformation $T^{-1}(\mathbf{z}(t_k))$. The reconstructed latent state is then mapped back to the physical space using the lifting operator introduced in Chapter 4 Eq. (4.15), yielding the estimated pedestrian density field. Table 6.5 reports the error norms of $\left\|r^{(1)}(\mathbf{x})\right\|_2$ evaluated on the testing dataset, measured in terms of the L_1, L_2, and L_∞ norms of the residual $\|r^{(1)}(\mathbf{x})\|_2$. For each norm, the statistics are computed over 50 independent training runs and summarized by the mean together with the 5th and 95th percentiles.

The results indicate consistently small residual errors across all norms. In particular, the mean values remain on the order of 10^{-3}, with very narrow spreads between the 5th and 95th percentiles. For instance, the L_1 error has a mean value of 2.21×10^{-3} with a percentile range of $(2.16E{-}03,\ 2.23E{-}03)$, while the L_2 and L_∞ norms exhibit similarly tight intervals.

The small magnitude of the residuals together with the narrow percentile gaps indicates that the PINN approximation reliably satisfies the functional equation (6.12) across the testing dataset and that the training procedure is robust with respect to initialization. Overall, these results demonstrate the stability and reproducibility of the learned transformation $T(\mathbf{x})$. Moreover, Fig. 6.10 depict the error across time for the reconstruction

Table 6.5: Testing dataset. Error norms, including L_1, L_2, and L_∞, quantify the residual $\left\|r^{(1)}(\mathbf{x})\right\|_2$ error. The analysis of error norms includes statistics such as the mean and 95th percentiles, aggregated over 50 runs.

$T(x_1, x_2)$	Error norm	PINN mean(5th,95th)
$T(x_1, x_2)$	$\|\cdot\|_1$	2.21E−03(2.16E−03, 2.23E−03)
	$\|\cdot\|_2$	1.59E−03(1.57E−03, 1.62E−03)
	$\|\cdot\|_\infty$	1.43E−03(1.41E−03, 1.46E−03)

of the density using the nonlinear observer. Initially, there exists a decrease drastically during the first 20 s as the POD reconstruction error baseline diminishes, and then slowly increase towards the end of the simulation due to error accumulation in the recursive forecasting. The observer-based reconstruction demonstrates remarkable approximation accuracy with the mean relative L_2 error remaining bounded below ∼8%(9%) by the end of the simulation horizon.

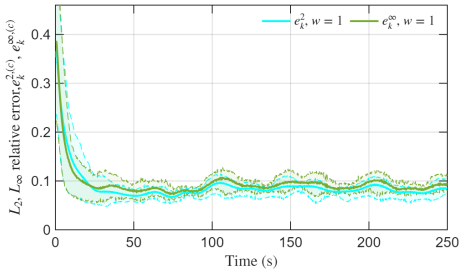
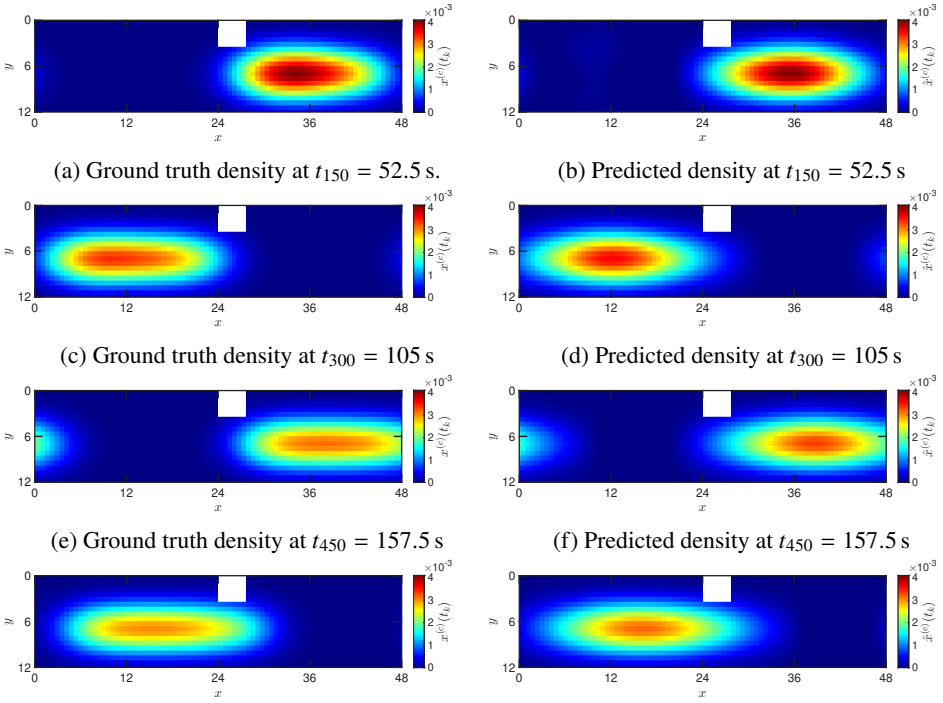


Figure 6.10: Relative errors with nonlinear observer.

For qualitative assessment, we directly compare the ground-truth normalized density fields $x^{(c)}(t_k)$ with the corresponding predictions $\hat{x}^{(c)}(t_k)$ obtained from the online deployment of the nonlinear observer. Figure 6.11 illustrates representative observations produced by the observer. The predicted densities preserve clearly the dominant spatial structure, propagation direction, and overall mass distribution of the ground truth, indicating that the reconstruction made with the nonlinear observer successfully captures the main macroscopic dynamics over long horizons.



(a) Ground truth density at $t_{150} = 52.5$ s.

(b) Predicted density at $t_{150} = 52.5$ s

(c) Ground truth density at $t_{300} = 105$ s

(d) Predicted density at $t_{300} = 105$ s

(e) Ground truth density at $t_{450} = 157.5$ s

(f) Predicted density at $t_{450} = 157.5$ s

(g) Ground truth density at $t_{600} = 210$ s

(h) Predicted density at $t_{600} = 210$ s

Figure 6.11: Comparison of the "ground truth" $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ normalized densities via the inversion of the nonlinear observer for the unidirectional flow case for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels (a), (c), (e), and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f), and (h), show the predicted one, for the time steps $t_{150} = 52.5$ s, $t_{300} = 105$ s, $t_{450} = 5$ s, and $t_{930} = 232.5$ s, respectively.

6.6 Discussion

The present chapter introduced a physics-informed machine learning framework for the construction of nonlinear discrete-time state observers. The methodology departs from classical multi-step observer design procedures [187, 188] by formulating the observer construction problem as the direct solution of a system of functional equations defining a nonlinear transformation operator.

Within this framework, the nonlinear system dynamics are reconstructed through a coordinate transformation that renders the transformed dynamics linear up to an output injection term. The existence and uniqueness of such a transformation are guaranteed under suitable structural assumptions, while its numerical realization is achieved through a PINN-based approximation scheme.

A key practical consideration is that the transformation operator may exhibit steep gradients or near-singular behavior in certain regions of the state space. This phenomenon was clearly illustrated in the benchmark studies, where singularities induce sharp variations in the transformation map. To address this issue, a greedy-wise training strategy based on a simple continuation procedure was adopted. By progressively expanding the training domain and reinitializing the network parameters, improved convergence behavior and numerical stability were achieved. Such continuation techniques not only enhance computational robustness but also provide insight into the structural properties of the transformation near singular regions.

From a numerical standpoint, the proposed PINN-based approach demonstrated improved approximation accuracy compared to classical power-series expansion methods, particularly in regions where higher-order expansions become unstable or computationally prohibitive. The PINN formulation allows the functional relation to be enforced directly over sampled regions of the state space, thereby avoiding the combinatorial growth of symbolic terms inherent to recursive Taylor constructions.

Most importantly, the deployment of this methodology in combination with the data-driven modeling framework for crowd dynamics demonstrates the practical feasibility of the proposed observer-based approach. The resulting observer evolves as an autonomous linear system in the transformed coordinates and does not require further access to the trained LSTM-based ROM during closed-loop simulations. This decoupling emphasizes the structural role of the learned transformation operator, which bridges the nonlinear latent dynamics and a linear estimation mechanism, thereby enabling efficient and reliable state reconstruction.

While the present implementation focuses on moderate-dimensional latent systems, scalability to higher-dimensional settings and robustness under stronger measurement noise remain open questions. In particular, the conditioning of the Jacobian matching constraint, the numerical stability of the inversion procedure, and the interaction between operator learning and observer design warrant further theoretical and computational

investigation.

These considerations highlight that the proposed framework should be viewed not merely as a numerical procedure, but as a structural approach to nonlinear state reconstruction in learned macroscopic systems. Its broader implications for data-driven modeling, estimation, and potentially control are explored in the concluding chapter.

7 Physics-Informed-based Control of Nonlinear Discrete-Time Dynamics

Abstract Chapters 3–6 established a data-driven framework for modeling multiscale complex systems, with particular application to crowd dynamics. Within this framework, the data-driven model was coupled with system-level tasks, such as state estimation, through nonlinear state reconstruction via observer linearization using Physics-Informed Machine Learning (PIML). The present chapter intendts to broad more in the system-level task that one can perform once we have a model to describe the dynamics of a given crowd in this case. The methodology proposed in this last chapter addresses the nonlinear feedback control problem in discrete time. We consider nonlinear input-driven systems and formulate the stabilization problem through exact feedback linearization and pole placement achieved in *a single step*. Rather than separating coordinate transformation and controller synthesis as classis and traditional methods often do, both components are constructed simultaneously by solving a system of nonlinear functional equations whose solution induces linear closed-loop dynamics with a prescribed eigenstructure. To compute the required nonlinear transformation in practice, thus a Physics-Informed Neural Network (PINN) formulation is introduced. The network is trained to satisfy the functional relations, equilibrium preservation constraints, and Jacobian matching conditions derived from local linearization. In the presence of steep gradients or near-singular behavior in the transformation map, a continuation-based greedy training strategy is employed to improve numerical stability and convergence. The proposed methodology is validated on benchmark nonlinear discrete-time systems, including cases with explicitly known governing equations as well as scenarios where only a black-box simulator is available. The results demonstrate accurate approximation of the transformation and successful closed-loop stabilization under the derived nonlinear feedback law. Although this chapter focuses primarily on methodological development and benchmark validation, it establishes the structural control layer required for future integration with the data-driven multiscale modeling framework developed throughout the thesis.

7.1 Nonlinear feedback control

Having addressed modeling and estimation, the natural completion of this framework is nonlinear feedback control. Within the discrete-time operator-based perspective adopted throughout this thesis, control must be formulated directly at the level of nonlinear evolution maps and must remain compatible with their structural properties.

We therefore consider nonlinear discrete-time input-driven dynamical systems of the form

$$\mathbf{x}(t_{k+1}) = f(\mathbf{x}(t_k), u(t_k)), \tag{7.1}$$

where $t_k \in \mathbb{N}$, $\mathbf{x}(t_k) \in \mathbb{R}^d$ denotes the state vector, $u(t_k) \in \mathbb{R}$ is the control input, and $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is assumed to be sufficiently smooth. Without loss of generality, the origin is assumed to be an equilibrium corresponding to zero input,

$$f(\mathbf{0}, 0) = \mathbf{0}. \tag{7.2}$$

If a non-zero equilibrium is considered, a translation of variables maps it to the origin.

The objective is local asymptotic stabilization of the equilibrium through state feedback. More precisely, we seek a nonlinear coordinate transformation

$$\mathbf{z} = T(\mathbf{x}), \tag{7.3}$$

which is locally invertible in a neighborhood of the equilibrium, together with a state-feedback control law of the form

$$u(t_k) = -c\,T(\mathbf{x}(t_k)), \tag{7.4}$$

where $c \in \mathbb{R}^{1\times d}$ is a constant row vector, such that the transformed closed-loop dynamics become linear,

$$\mathbf{z}(t_{k+1}) = A\mathbf{z}(t_k), \tag{7.5}$$

with $A \in \mathbb{R}^{d\times d}$ a design matrix whose eigenvalues lie strictly inside the unit disc.

In contrast to classical feedback linearization procedures that separate coordinate transformation and pole placement into successive design steps, the approach adopted here achieves both objectives simultaneously. The nonlinear transformation $T(x)$ and the feedback law are constructed as the solution of a system of nonlinear functional equations encoding the requirement that the closed-loop dynamics of (7.1) match the linear target system (7.5) in the transformed coordinates.

Under appropriate controllability and spectral separation conditions, these functional equations admit a unique locally analytic and invertible solution in a neighborhood of the equilibrium. The resulting transformation establishes a structural correspondence between the nonlinear dynamics in the original coordinates and linear closed-loop behavior in the transformed coordinates.

From a computational perspective, however, explicit analytic expressions for the transformation map are rarely available, especially in moderately high-dimensional systems or when only simulator access to f is assumed. The subsequent sections therefore introduce a PINN-based formulation for approximating the transformation by directly enforcing the defining functional equations together with equilibrium preservation and Jacobian matching constraints.

7.2 Single-step feedback linearization via nonlinear functional equations

To achieve simultaneous feedback linearization and pole placement, we seek a nonlinear transformation and feedback law that induce prescribed linear closed-loop dynamics. Substituting the state feedback law

$$u(t_k) = -c\,T(\mathbf{x}(t_k)) \tag{7.6}$$

into the nonlinear system (7.1) yields the closed-loop dynamics

$$\mathbf{x}(t_{k+1}) = f(\mathbf{x}(t_k), -c\,T(\mathbf{x}(t_k))). \tag{7.7}$$

The requirement that the transformed state $\mathbf{z} = T(\mathbf{x})$ satisfies the linear dynamics

$$\mathbf{z}(t_{k+1}) = A\mathbf{z}(t_k) \tag{7.8}$$

leads to the following system of nonlinear functional equations (NFEs):

$$T\big(f(\mathbf{x}, -c\,T(\mathbf{x}))\big) = A\,T(\mathbf{x}), \tag{7.9}$$

together with the equilibrium condition

$$T(\mathbf{0}) = \mathbf{0}. \tag{7.10}$$

The above system encodes both feedback linearization and pole placement in a single step. The existence of a solution to (7.9) is guaranteed under suitable structural assumptions.

Theorem 7.1. *Consider the nonlinear discrete-time system* (7.1) *and the associated system of nonlinear functional equations* (7.9) *with initial condition* $T(\mathbf{0}) = \mathbf{0}$. *Let*

$$J = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, 0), \qquad G = \frac{\partial f}{\partial u}(\mathbf{0}, 0) \neq 0. \tag{7.11}$$

Assume the following conditions:

Assumption 1 (Local controllability). The controllability matrix

$$C = \begin{bmatrix} G & JG & \dots & J^{d-1}G \end{bmatrix} \tag{7.12}$$

has full rank, i.e.,

$$\operatorname{rank}(C) = d. \tag{7.13}$$

Assumption 2 (Poincaré domain). The eigenvalues f_i of the matrix A lie strictly inside the unit disc.

Assumption 3 (Spectral separation). The eigenspectra of A and J are disjoint,

$$\sigma(A) \cap \sigma(J) = \emptyset. \tag{7.14}$$

Assumption 4 (Non-resonance condition). The eigenvalues f_i of A and λ_j of J do not satisfy relations of the form

$$\prod_{i=1}^{d} f_i^{m_i} = \lambda_j, \tag{7.15}$$

for non-negative integers m_i satisfying

$$\sum_{i=1}^{d} m_i > 0. \tag{7.16}$$

Assumption 5 (Observability of (c, A)). The matrix

$$O = \begin{bmatrix} c \\ cA \\ \vdots \\ cA^{d-1} \end{bmatrix} \tag{7.17}$$

has full rank, i.e.,

$$\operatorname{rank}(O) = d. \tag{7.18}$$

Then, the system of nonlinear functional equations (7.9) admits a unique locally analytic and locally invertible solution $T(\mathbf{x})$ in a neighborhood of the origin. The simultaneous implementation of the nonlinear transformation $\mathbf{z} = T(\mathbf{x})$ and the feedback control law $u = -cT(\mathbf{x})$ induces the linear closed-loop dynamics

$$\mathbf{z}(t_{k+1}) = A\mathbf{z}(t_k), \tag{7.19}$$

whose eigenvalues coincide with those of the design matrix A.

The equilibrium condition $T(\mathbf{0}) = \mathbf{0}$ ensures preservation of equilibrium properties under the coordinate transformation. Since the eigenvalues of A lie inside the unit disc,

the closed-loop system is locally asymptotically stable in the Lyapunov sense. The invertibility of T guarantees that stabilization in the transformed coordinates implies stabilization of the original nonlinear system.

To further characterize the transformation, we examine the linearization of (7.1) around the equilibrium $(\mathbf{0}, 0)$:

$$\mathrm{d}\mathbf{x}(t_{k+1}) = J\,\mathrm{d}\mathbf{x}(t_k) + G\,\mathrm{d}u(t_k). \tag{7.20}$$

Under the feedback law

$$\mathrm{d}u(t_k) = -c\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\,\mathrm{d}\mathbf{x}(t_k), \tag{7.21}$$

the linearized closed-loop dynamics become

$$\mathrm{d}\mathbf{x}(t_{k+1}) = \big(J - Gc\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\big)\mathrm{d}\mathbf{x}(t_k). \tag{7.22}$$

On the other hand, linearization of the transformed system yields

$$\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\,\mathrm{d}\mathbf{x}(t_{k+1}) = A\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})\,\mathrm{d}\mathbf{x}(t_k). \tag{7.23}$$

Combining the above expressions leads to the matrix equation

$$\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})J - A\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0}) = \frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})Gc\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0}), \tag{7.24}$$

which serves as a phase (Jacobian matching) condition for the transformation at the equilibrium. Equation (7.24) plays a fundamental role in the numerical construction of $T(x)$, as it provides the local linear constraint that must be satisfied in conjunction with the nonlinear functional equations.

This proposed PIML scheme can be easily implemented. In fact, for our illustrations, we developed a "home-made" code in Matlab 2022; for training, we wrapped around it the Levenberg-Marquard optimization algorithm as implemented by the `nonlinsq` function. For comparison purposes, an implementation in Python using the Keras API of TensorFlow library was also performed. To demonstrate the efficiency of the proposed continuation/greedy PIML scheme, we used a benchmark two-dimensional discrete-time model whose feedback-linearizing control law can be derived analytically [189]. The particular transformation map (and thus the attendant feedback-linearizing control law) exhibits a singularity at the domain boundary, thus making it difficult to approximate it numerically close to that point, especially through a power-series expansion. For illustration purposes, we also compared the numerical approximation accuracy of the proposed PIML greedy scheme against the standard power-series expansion, as well as Matlab and Python's TensorFlow-based implementations with automatic differentiation that was used to learn the transformation in the entire domain. Furthermore, we considered two different scenarios, namely: (a) one where we assumed that the equations of the model are explicitly known, and (b) one where we assumed that only a black-box simulator is available, i.e., pertinent equations are not available explicitly in closed form.

7.3 Approximation of the transformation via PINNs

Although Theorem 7.1 guarantees the local existence and uniqueness of an analytic solution to the nonlinear functional equation system Eq. (7.9), explicit analytic expressions for the transformation map $T(\mathbf{x})$ are rarely available in practice. This difficulty becomes particularly pronounced in moderately high-dimensional systems or when the function $f(\mathbf{x}, u)$ is accessible only through simulation.

To compute an approximation of the transformation map, we adopt a physics-informed neural network (PINN) formulation. The central idea is to approximate $T(\mathbf{x})$ by a neural network $\hat{T}(\mathbf{x})$ and enforce the nonlinear functional equation Eq. (7.9), the equilibrium condition, and the Jacobian matching constraint Eq. (7.24) through a residual-based loss function.

7.3.1 Neural network architecture

We approximate the transformation $T(\mathbf{x})$ by a feedforward neural network with two hidden layers and a linear output layer. Denoting the approximation by $\hat{T}(\mathbf{x})$, its j-th component is given by

$$\hat{T}_j(\mathbf{x}) = \sum_{i=1}^{N_2} W_{ij}^{(o)} \phi_i^{(2)} \left(\sum_{s=1}^{N_1} W_{si}^{(2)} \phi_s^{(1)} \left(\sum_{k=1}^{d} W_{ks}^{(1)} x_k + \beta_s^{(1)} \right) + \beta_i^{(2)} \right) + \beta_j^{(o)}. \tag{7.25}$$

Equivalently, in compact matrix form:

$$\hat{T}(\mathbf{x}) = {W^{(o)}}^T \Phi_2 \big({W^{(2)}}^T \Phi_1 ({W^{(1)}}^T \mathbf{x} + \beta^{(1)}) + \beta^{(2)} \big) + \beta^{(o)}. \tag{7.26}$$

Here:

- $W^{(1)}, W^{(2)}, W^{(o)}$ are weight matrices,
- $\beta^{(1)}, \beta^{(2)}, \beta^{(o)}$ are bias vectors,
- Φ_1 and Φ_2 collect the activation functions of the hidden layers.

The set of trainable parameters is denoted by

$$\theta = \left(W^{(o)}, W^{(2)}, W^{(1)}, \beta^{(o)}, \beta^{(2)}, \beta^{(1)} \right). \tag{7.27}$$

7.3.2 Loss function formulation

A schematic representation of the proposed PINN-based feedback linearization framework is shown in Fig. 7.1. The neural network approximation $\hat{T}(\mathbf{x})$ is trained to satisfy the nonlinear functional Eq. (7.9), the equilibrium constraint, and the Jacobian matching condition Eq. (7.24) over the domain of interest.

The residual enforcing the nonlinear functional equation is defined as

$$r_{ij}^{(1)}(\mathbf{x}_i, \hat{T}(\mathbf{x}_i)) = \hat{T}_j\big(f(\mathbf{x}_i, -c\hat{T}(\mathbf{x}_i))\big) - \alpha_j \hat{T}(\mathbf{x}_i), \tag{7.28}$$

where α_j denotes the j-th row of matrix A.

Equilibrium preservation is enforced through the residual

$$r_j^{(2)} = \hat{T}_j(\mathbf{0}), \tag{7.29}$$

while the Jacobian matching condition is imposed through

$$r_{jk}^{(3)} = \frac{\partial \hat{T}_j}{\partial x_k}(\mathbf{0}) - \frac{\partial T_j}{\partial x_k}(\mathbf{0}), \tag{7.30}$$

The Jacobian $\frac{\partial T}{\partial \mathbf{x}}(\mathbf{0})$ is obtained by solving the matrix equation (7.24).

7.3.3 Gradient computation

Assuming sufficient smoothness of the loss function, its gradient with respect to any parameter $\theta_\ell \in \theta$ is

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_\ell} = \sum_{i=1}^{M}\sum_{j=1}^{d} r_{ij}^{(1)} \frac{\partial r_{ij}^{(1)}}{\partial \theta_\ell} + \sum_{j=1}^{d} r_j^{(2)} \frac{\partial r_j^{(2)}}{\partial \theta_\ell} + \sum_{j=1}^{d}\sum_{k=1}^{d} r_{jk}^{(3)} \frac{\partial r_{jk}^{(3)}}{\partial \theta_\ell}. \tag{7.31}$$

For the nonlinear functional residual,

$$\frac{\partial r_{ij}^{(1)}}{\partial \theta_\ell} = \frac{\partial \hat{T}_j(f(x_i, -c\hat{T}(x_i)))}{\partial \theta_\ell} \frac{\partial f(x_i, -c\hat{T}(x_i))}{\partial u} \left(-c\frac{\partial \hat{T}(x_i)}{\partial \theta_\ell}\right) - \alpha_j^T \frac{\partial \hat{T}(x_i)}{\partial \theta_\ell}. \tag{7.32}$$

All required derivatives of $\hat{T}$ with respect to inputs and parameters can be computed either analytically from (7.26) or via automatic differentiation when using modern computational frameworks. For completeness, the analytical expressions of the derivatives involved in the gradient computation are reported in Appendix G.

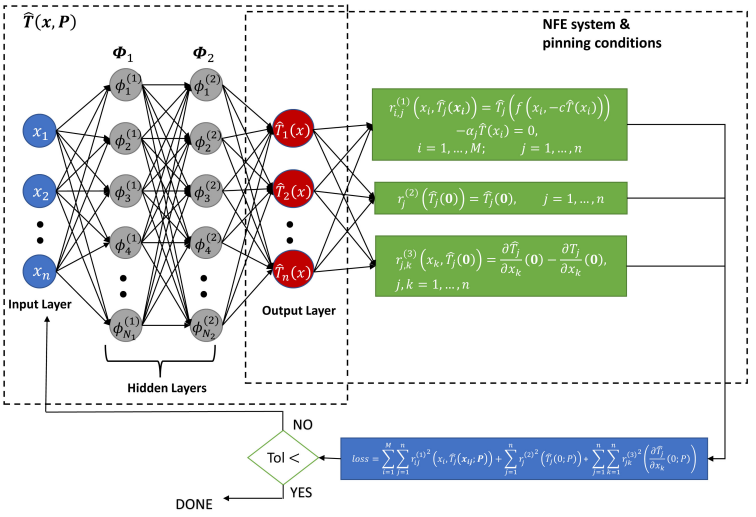


Figure 7.1: A schematic of the PIML scheme for the feedback linearization of discrete-time systems.

7.3.4 Greedy continuation strategy

A practical difficulty arises when the transformation map exhibits steep gradients or near-singular behavior within the domain D. Direct training over the entire domain may lead to poor convergence.

To address this issue, we employ a continuation-based greedy strategy. Consider a nested sequence of subdomains

$$D_1 \subset D_2 \subset \cdots \subset D_k \subset \cdots \subset D_d = D. \tag{7.33}$$

The PINN is first trained in D_1, a small neighborhood around the equilibrium. Upon convergence, the learned parameters P_1 are used as initialization for training in D_2, i.e.,

$$P_k^{(0)} = P_{k-1}. \tag{7.34}$$

This zero-order continuation procedure enhances stability and facilitates convergence in regions where the transformation varies rapidly.

The methodology described above applies both when the governing equations $f(x, u)$ are explicitly known and when only a black-box simulator is available. In the latter case, required derivatives can be computed numerically or through automatic differentiation, depending on implementation.

This PINN-based construction provides a computational realization of the nonlinear transformation guaranteed by Theorem 7.1, enabling simultaneous feedback linearization and pole placement in systems where analytic solutions are not accessible.

7.4 The benchmark problem

To evaluate the performance of the methods proposed in this work, we have considered the following nonlinear discrete-time system [189]:

$$\begin{aligned} x_1(t+1) &= exp(0.3x_2(t))\sqrt{(1+x_1(t)+x_2(t))} - 1 - 0.4x_2(t) + 0.5u(t) \\ x_2(t+1) &= 0.5\ln(1+x_1(t)+x_2(t)) + 0.4x_2(t) \end{aligned} \tag{7.35}$$

The Jacobian matrix of the above system at the equilibrium $(\mathbf{0}, 0)$ is $\frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, 0) = \begin{bmatrix} 0.5 & 0.4 \\ 0.5 & 0.9 \end{bmatrix}$, and its eigenvalues $\lambda_1 = 0.2101$ and $\lambda_2 = 1.1899$. The matrix A is chosen to be $A = \begin{bmatrix} 0.5 & 0.3 \\ 0.5 & 0.4 \end{bmatrix}$, with eigenvalues $k_1 = 0.8405$ and $k_2 = 0.0595$. Due to the choice of matrix A, its eigenvalues are not related to the eigenvalues of the Jacobian matrix $\frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, 0)$ through any equations of the type Eqs. (6.13)-(6.14). Moreover, the following row vector c was chosen:

$$c = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{7.36}$$

Please notice, that all conditions of Theorem 2.1. are met by the system Eq. 7.35, and therefore the associated system of NFEs (3):

$$\begin{aligned} T_1(exp(0.3x_2(t))\sqrt{(1+x_1(t)+x_2(t))} - 1 - 0.4x_2(t) - 0.5T_1, \\ 0.5\ln(1+x_1(t)+x_2(t)) + 0.4x_2(t)) = 0.5T_1 + 0.3T_2 \\ T_2(exp(0.3x_2(t))\sqrt{(1+x_1(t)+x_2(t))} - 1 - 0.4x_2(t) - 0.5T_1, \\ 0.5\ln(1+x_1(t)+x_2(t)) + 0.4x_2(t)) = 0.5T_1 + 0.4T_2 \\ T_1(0,0) = 0 \\ T_2(0,0) = 0, \end{aligned} \tag{7.37}$$

admits a unique locally analytic and invertible solution around the equilibrium point $(x_1, x_2) = (0, 0)$. Indeed, please notice that

$$\frac{\partial T}{\partial x}(0,0) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{7.38}$$

with a $det[T] \neq 0$, results in a locally invertible around the equilibrium point $(x_1, x_2) = (0, 0)$ solution, that can be also calculated analytically in closed-form [189]:

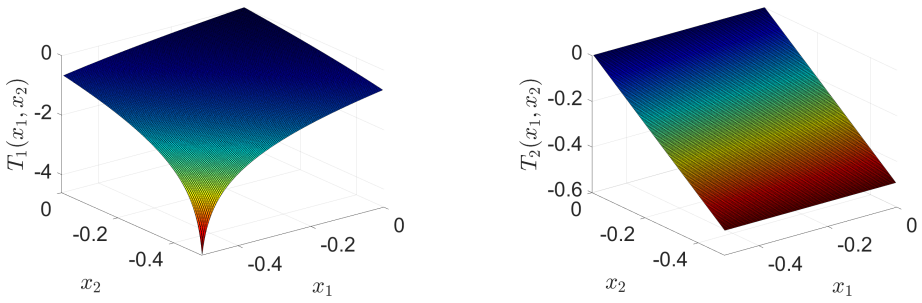
$$T_1(x_1, x_2) = \ln(1 + x_1 + x_2), \quad T_2(x_1, x_2) = x_2. \tag{7.39}$$

In agreement with Theorem 2.1, the proposed feedback-linearizing and pole-placing nonlinear feedback control law can be explicitly written as follows:

$$u = -cT(\mathbf{x}) = -\ln(1 + x_1 + x_2). \tag{7.40}$$

Our benchmark problem encompasses a set of singularities of the nonlinear transformation when $x_1 + x_2 = -1$. Here, we sought to learn the feedback-linearizing control law in the domain $[x_L, 0] \times [x_L, 0] = [-0.495, 0] \times [-0.495, 0]$.

Figure (7.2) depicts the analytical solutions $T_1(x_1, x_2), T_2(x_1, x_2)$, of the associated system of NFEs. We note that it encompasses the solution domain, which has been deliberately



(a) (b)

Figure 7.2: Analytical solution of the NFEs (7.37) in $[-0.495, 0] \times [-0.495, 0]$. (a) $T_1(x_1, x_2) = \ln(1 + x_1 + x_2)$. A steep-gradient at $(-0.495, -0.495)$ is due to the presence of a singular point at $(x_1, x_2) = (-0.5, -0.5)$. (b) $T_2(x_1, x_2) = x_2$.

.

chosen to be $x_1, x_2 \in [-0.495, 0]$ since $T_1(x_1, x_2)$ exhibits a singular point at $(x_1, x_2) = (-0.5, -0.5)$. The first reasonable step is, therefore, to verify by comparing with the analytical solution, the numerical approximation accuracy of the proposed PIML scheme in this region and assess its impact on the performance profile of the resulting feedback-linearizing control law.

7.5 Numerical results

We present the numerical results in two subsections. In Eq. 7.4, we consider the case where the discrete nonlinear model Eq. (7.1) is assumed to be explicitly available. In this setting, the necessary derivatives entering the optimization algorithm can be computed analytically, as described in Section 7.3, or by exploiting automatic differentiation toolkits.

Within the greedy PIML framework introduced in Section 7.3, we implemented two computational schemes: (a) a home-made code in MATLAB, using the Levenberg–Marquardt algorithm for training; and (b) an implementation using the Keras API of the TensorFlow library in Python, employing the BFGS optimization algorithm. The latter scheme utilizes automatic differentiation by default to compute the required gradients.

To assess the performance of the greedy continuation strategy, we also report results

obtained when the PIML model is trained over the entire domain at once, without domain expansion. This comparison highlights the impact of the continuation procedure on convergence behavior and approximation accuracy.

In Subsection 7.5.2, we present the results under the assumption that the discrete nonlinear model Eq. (7.1) is not explicitly available and only a black-box simulator is accessible. In this setting, the derivatives required by the optimization procedure are approximated numerically using centered finite differences.

For training purposes using the greedy approach, we considered 20 equispaced distributed collocation points for each one of the two inputs x_1 and x_2, i.e. we used a grid of 20×20 points equispaced distributed for each step of the greedy approach. Different (denser) sizes of the grid (e.g. using a grid of 100×100 points) with more neurons for each layer did not affect qualitatively the results and corresponding performances, as we also show. Thus, for the greedy-approach, we started by considering a grid in $[-0.2, 0] \times [-0.2, 0]$. Then with a step of -0.05, we used as initial guesses of the unknown weights and biases of the PIML the ones obtained from the training procedure from the previous grid, and repeated training until the interval $[-0.45, 0] \times [-0.45, 0]$ was reached. From this interval and on, we optimized iteratively the PIML using progressively bigger intervals with a step of -0.01, for each of the two inputs x_1 and x_2, up to the interval $[-0.49, 0] \times [-0.49, 0]$. After this interval we augmented progressively the grid with steps of -0.001 until the entire domain of interest, $[-0.495, 0] \times [-0.495, 0]$, was reached. As mentioned before, additionally, we have also used the TensorFlow library to learn the transformation law both with the greedy approach and in the entire domain $[-0.495, 0] \times [-0.495, 0]$. The performances of the PIML schemes were also compared with a $6th$ order power-series expansion of $T_1(x_1, x_2), T_2(x_1, x_2)$, thus resulting in equivalent, to the PIML, number of unknowns.

For testing purposes, we used the roots of the Chebyshev polynomial of the first kind of degree k. For our illustrations, we selected a grid of 50×50 Chebyshev points for each of the intervals in $[-0.495, 0] \times [-0.495, 0]$. In particular, in the interval $[a, b]$ with $a, b \in \mathbb{R}$, the grid was created using the following Chebyshev collocation points:

$$x_n = \frac{1}{2}(a + b) - \frac{1}{2}(a - b)cos\left(\frac{2n - 1}{2n}\pi\right), \quad n = 1, \cdots, h \tag{7.41}$$

with $a = -0.495$ and $b = 0$, and therefore equation (7.41) can now be expressed as follows:

$$x_n = \frac{1}{2}(-0.495) - \frac{1}{2}(-0.495)cos\left(\frac{2n - 1}{2n}\pi\right), \quad n = 1, \cdots, h \tag{7.42}$$

7.5.1 The case of the explicitly available model

First, we have expanded both the right-hand-side of the model (7.1) and the nonlinear transformation $T(x_1, x_2)$ to a $6th$ order power-series, and equated same order terms up to the $6th$ order of $T(f(\mathbf{x}, -cT(\mathbf{x})))$ and $AT(\mathbf{x})$ on both sides of the associated NFEs (see Appendix G.1), thus building a system of algebraic equations that can be solved for the coefficients of the $6th$ order polynomial approximation. For this task, we used Matlab's symbolic toolbox.

For our PIML scheme, we constructed an ANN with two hidden layers, fully connected, with five neurons in each layer. More specifically, for both the home-made Matlab code and the TensorFlow (TF) implementation, the activation function was chosen to be the $sigmoid(\cdot)$ function. For the training, in the Matlab implementation, we used the function *lsqnonlin*, which implements the Levenberg-Marquart (LM) algorithm. In the LM scheme, we have set as stopping criterion a threshold of FuncTol = 10^{-12}. Moreover, we have set a maximum number of $100,000$ iterations and a maximum number of function iterations equal to $12,000$. Finally, to initialize the weights and biases, we have used uniformly distributed random numbers in the interval $[0, 1]$ by employing the *rand* function of Matlab.

For our computations, we have also used the Keras API of TensorFlow. Thus, we have used automatic differentiation (AD) [190] to find the required derivatives for the optimization process. With the TensorFlow, we have tried different optimizers; namely the Stochastic gradient descent (SGD), the ADAM optimizer and the BFGS optimizer. For the SGD and ADAM optimizers, we have used the default values, except for the learning rate which has been selected using a piecewise constant decay varying from 10^{-2} to 10^{-4} as the number of epochs increased. We have chosen $100,000$ epochs since using more epochs did not seem to considerably improve the results derived, whereas for the BFGS we have used the library *tensorflow probability*. Furthermore, we have used a maximum number of iterations equal to $100,000$, and a stopping condition of FuncTol = 10^{-16}. The best results obtained with the TF were derived with the aid of BFGS, and these are the results presented in our comparative assessment, shown below. For clarity, we report here only the results obtained on the test sets, while the corresponding results for the training sets are provided in Appendix G.4.

Figure (7.3) depicts the performance of the schemes tested on a Chebyshev grid. Fig. 7.3 reports the numerical approximation accuracy, defined as the difference between the computed and analytical solutions, obtained by the different schemes on the training set. Panels (a)–(b) show the results obtained using a 6th-order power-series expansion of the nonlinear transformation and the right-hand side of the discrete model. As expected, this approach yields zero error for the $T_2(x_1, x_2)$ component and good accuracy for $T_1(x_1, x_2)$ only in a neighborhood of the linearization point $[0, 0]$. Away from this region, however, the approximation deteriorates significantly, reaching errors of order 10^0 near the edge of the grid close to the singular point. Panels (c)–(d) display the results obtained with the PIML framework implemented in TensorFlow using the BFGS optimizer to learn

the nonlinear transformation over the entire domain. In this case the approximation accuracy remains poor, particularly near the singularity, with errors again of order 10^0. Panels (e)–(f) show the results obtained with the same TensorFlow implementation when a greedy training strategy is employed, while panels (g)–(h) report the results obtained using a MATLAB implementation with the LM optimizer combined with the greedy procedure. The greedy training strategy significantly improves the approximation accuracy compared with training over the entire domain at once. In particular, the TensorFlow-based PIML achieves errors of order 10^{-2}, while the MATLAB implementation attains errors of order 10^{-3} near the edge point $(-0.495, -0.495)$. The results are similar for the training and the test set. Table 7.1 detail the numerical approximation accuracy of the various schemes fo the test sets.

Table 7.1: Model explicitly available. Test sets (grids of 50×50 Chebyshev-distributed points). Error norms (L_1, L_2 and L_∞) between the analytical and computed solution of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ using the various schemes trained both greedy and in the entire domain $[-0.495, 0] \times [-0.495, 0]$.

Transformation $T(x)$	Error norm	power-series $6th$ order	PIML(TF) Entire domain	PIML(Matlab) Greedy	PIML(TF) Greedy
$T_1(x_1, x_2)$	$\|\cdot\|_1$	6.89E+01	2.17E+01	3.40E−02	4.77E−02
	$\|\cdot\|_2$	4.55E+00	1.44E+01	2.63E−03	4.70E−02
	$\|\cdot\|_\infty$	2.88E+00	1.00E+01	1.41E−03	2.60E−02
$T_2(x_1, x_2)$	$\|\cdot\|_1$	0	2.87E+00	1.45E−02	1.89E−01
	$\|\cdot\|_2$	0	1.97E+00	1.55E−03	1.84E−01
	$\|\cdot\|_\infty$	0	1.23E+00	7.62E−04	1.28E−01

7.5.2 The black-box simulator case

As opposed to the "explicitly known model" PIML scheme, where we used analytical derivatives, in the black-box simulator scheme, the derivatives were estimated using central finite differences with a perturbation step of $eps = 10^{-4}$. Here, for illustration purposes, we have implemented the PIML only on Matlab in a "fully numerical way". Once more, for clarity the results regarding the training set can be found in Appendix G.4.

Fig. 7.4 reports the numerical approximation accuracy obtained on the test set for the different approximation schemes. Panels (a)–(b) correspond to the power-series expansion, which yields relatively poor accuracy near the singular point $[-0.495, -0.495]$, with errors of order 10^0. Panels (c)–(d) show the results obtained with the PIML scheme implemented in Matlab and trained over the entire domain, where the approximation error reduces to approximately 10^{-1} in the vicinity of the singularity. Panels (e)–(f) present the results obtained using the same PIML framework combined with the greedy training procedure. In this case, the approximation error in the region characterized by steep gradients decreases to the order of 10^{-3}, clearly outperforming both the power-series expansion and the PIML model trained over the entire domain at once. Moreover, the

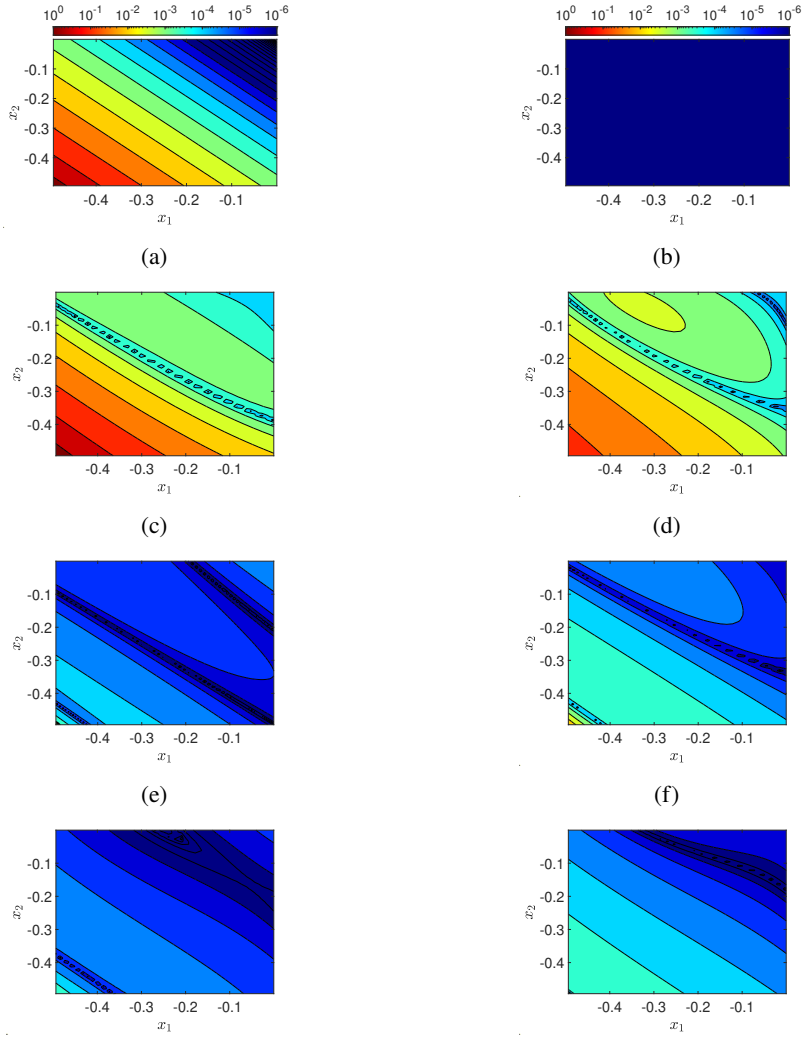


Figure 7.3: Model explicitly available. Test sets (grids of 50×50 Chebyshev-distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model Eq. (7.35) in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PIML in Tensorflow trained in the entire domain $[-0.495, 0] \times [-0.495, 0]$. (e),(f) PIML in Tensorflow trained via the greedy approach. (g),(h) PIML in Matlab trained via the greedy approach.

numerical accuracy achieved near the singular point $[-0.495, -0.495]$ is comparable to that obtained when the underlying model is assumed to be explicitly known. Table 7.2 detail the numerical approximation accuracy of the various schemes, for test sets.

Finally, in Fig. (7.5), we depict the numerical approximation error of the transformation map $T_1(x_1, x_2)$ in terms of the L_2 norm with respect to the size of the domain for four different schemes, namely: (a) a PIML implemented in TensorFlow, trained in the entire domain (blue line), (b) a PIML implemented on Matlab trained with the Levenberg-Marquardt in the entire domain (orange line), (c) a PIML implemented on Matlab trained with the Levenberg-Marquardt using the greedy procedure (yellow line), and, (d) a PIML implemented in TensorFlow using the greedy procedure (purple line). For the construction of the diagram, we have used a grid of 20×20 equispaced distributed collocation points. Starting with a -0.2×-0.2 grid and using a step of -0.05 we performed the training process each time until the interval -0.45×-0.45 was reached. Then, for the interval between -0.45×-0.45 and -0.49×-0.49, the step chosen for augmenting the grid were of size -0.01 and finally from this last interval to -0.495×-0.495 the step chosen was of size -0.001.

It should be pointed out that in the present study, conducted in the discrete time domain, a numerical simulation-based approach is proposed through which the attainment of sufficiently small and bounded testing errors and closed-loop stability can be verified.

Table 7.2: Black-box simulator. Test sets (grids of 50×50 Chebyshev-distributed points). Error norms (L_1, L_2 and L_∞) between the analytical and computed solution of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ using the various schemes trained both greedy and in the entire domain $[-0.495, 0] \times [-0.495, 0]$.

Transformation $T(x)$	Error norm	power-series $6th$ order	PIML(Matlab) Entire domain	PIML(Matlab) Greedy
$T_1(x_1, x_2)$	$\|\cdot\|_1$	2.17E+01	3.54E+00	7.35E−03
	$\|\cdot\|_2$	1.84E+01	2.65E+00	7.18E−03
	$\|\cdot\|_\infty$	4.89E+00	1.81E+00	4.36E−03
$T_2(x_1, x_2)$	$\|\cdot\|_1$	1.19E+00	7.87E−01	1.77E−02
	$\|\cdot\|_2$	1.05E+00	4.84E−01	1.64E−02
	$\|\cdot\|_\infty$	9.03E−01	2.98E−01	9.18E−03

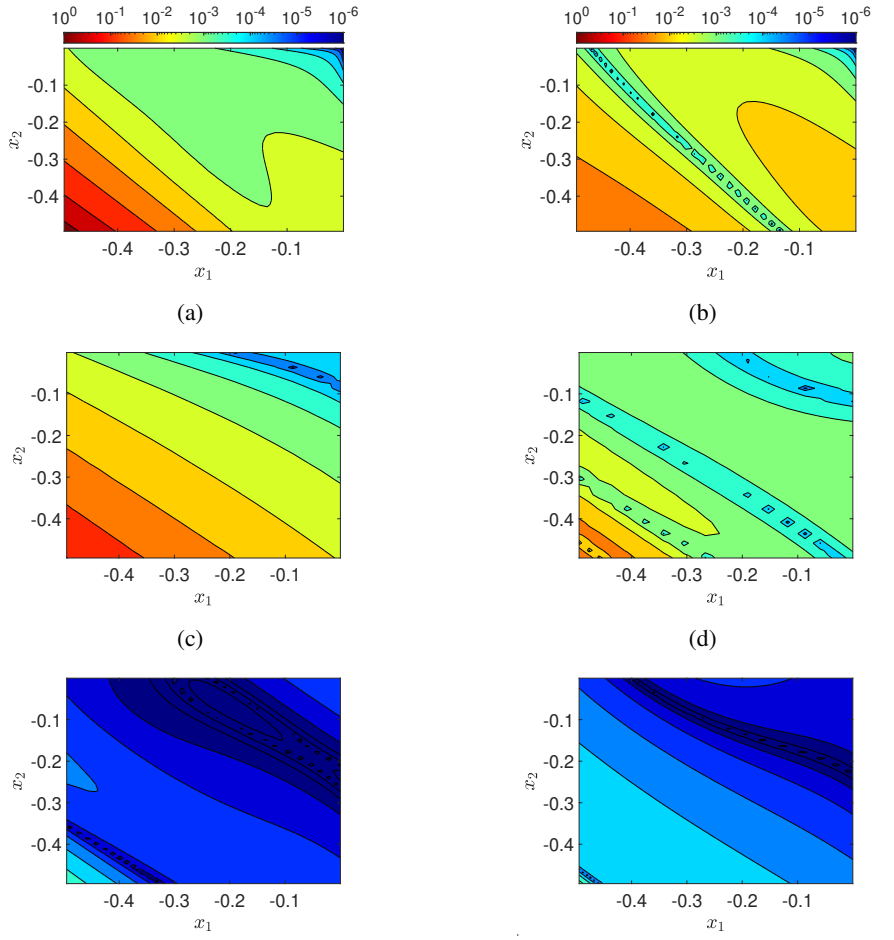


Figure 7.4: Black-box simulator. Test sets (grids of 50×50 Chebyshev-distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PIML in Matlab trained in the entire domain $[-0.495, 0] \times [-0.495, 0]$. (e),(f) PIML in Matlab trained via the greedy approach.

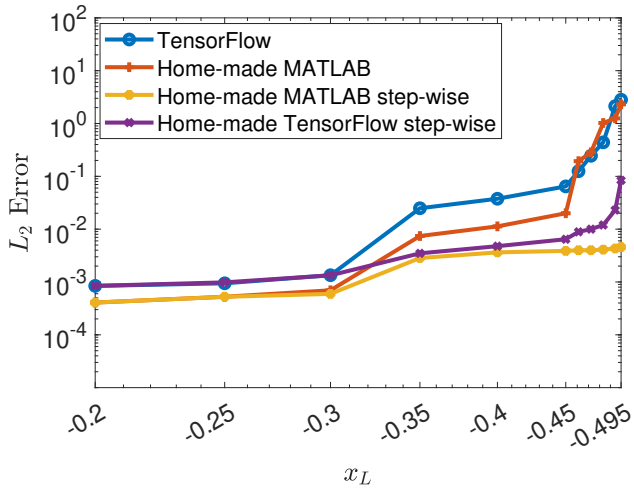


Figure 7.5: L_2 norm of the numerical approximation accuracy between the analytical transformation law $T_1(x_1, x_2)$ (see Eq. 7.39) and the various PIML implementations trained both with the greedy approach and in the entire domain, for various sizes of the domain, $[x_L, 0] \times [x_L, 0]$. For our illustrations, we have used grids of 20×20 equispaced distributed collocation points.

.

7.6 Discussion

We proposed a PIML-based scheme for the single-step feedback linearization with pole placement for nonlinear discrete-time systems. Here, we considered a system for which the linearizing transformation map and state feedback control law exhibit a singular point, and thus very steep transformation gradients near it. As the underlying optimization problem may lead to a poor solution (for example due to the effect of random initialization of the weights of the PIML), we have chosen to implement a greedy approach, thus tessellating the "hard" (in the entire domain) training/optimization problem into a sequence of simpler ones; this has also been suggested in other studies (see e.g. [191, 192]). Such a greedy training strategy, used to initialize weights in a region near a good local minimum, facilitates the optimization algorithm by implicitly acting as a regularization technique and thus resulting in a better generalization [191]. The existence of a singularity, on and beyond which the feedback linearization fails, is a hallmark of many problems that seek useful transformation by formulating and solving functional differential equations [193]. The same type of issue will, for example, arise in trying to compute flow-box transformations [194], or transformations to linearity (in the Koopman operator context [195, 196]). Understanding how to test for such singularities, adaptively re-mesh in their neighborhood, estimate the associated singularity exponents, and even considering possible analytic continuations beyond them, is an important issue [193]. In fact, here we

have implemented a simple zero-th order continuation in order to provide better initial guesses for the unknown weights of the PIML scheme to regions close to the singularity. In future work, we will aim at exploiting more advanced continuation techniques, such as the natural continuation technique proposed in Fabiani et al. [192] for providing analytically initial guesses for the unknown weights of random projection networks for the solution of stiff ODEs and index-1 DAEs containing steep gradients in their solution profiles, or arc-length continuation for tracing branches of solutions and approximation of manifolds up to or even beyond critical/singular points (see for example [197, 198]). Thus bridging ML programming techniques with concepts from continuation techniques, have the potential to significantly facilitate computational experimentation and learning, and thus assist in the study of such singularities, possibly suggesting approaches to their mitigation.

Importantly, it should be pointed out that a theoretically rigorous establishment of closed-loop stability under feedback action, considering the approximation errors introduced by machine learning algorithms, remains a challenging research endeavor. The latter requires a creative integration of conceptual and methodological tools derived from machine learning theory, control theory and uncertainty quantification, as suggested in [199, 200, 201, 202, 54]. Indeed, in the aforementioned studies closed-loop stability is guaranteed under a certain set of assumptions/conditions relying on: i) sufficiently small and bounded errors induced by the underlying neural network architecture [199, 200, 201, 202], or the uncertainty quantification of the error bounds from the solution of the pre-image problem when manifold learning techniques are used [54], ii) more sophisticated generalization error bounds using statistical machine learning theory, as well as iii) stochastic formulations of the stability problem under consideration. For a general review of the problem of uncertainty quantification in scientific machine learning, the interested reader is referred to a recent study presented in [203].

Finally, the application of this methodology within a data-driven modeling framework for crowd dynamics represents a natural continuation of this thesis. This chapter establishes the theoretical and methodological foundations for such integration, while its implementation is left for future work.

8 Conclusions and future work

This thesis work addressed the fundamental problem of bridging microscopic agent-based descriptions and macroscopic system-level models in complex multi-agent systems, with a primary focus on crowd dynamics. In many real-world settings, explicit macroscopic governing equations are unavailable or prohibitively difficult to derive, making traditional continuum modeling approaches insufficient for predictive analysis, estimation, and control. The work presented in this thesis proposed an alternative perspective based on a new generation equation-free formulation [123], in which macroscopic state representations and their governing evolution operators are identified directly from microscopic simulation data.

The central methodological contribution of the thesis is the development of an end-to-end computational pipeline linking microscopic simulations, macroscopic observable construction, reduced-order state representations, data-driven identification of macroscopic evolution operators, and their integration with estimation and control methodologies. Starting from agent-based simulations of pedestrian dynamics, macroscopic density fields were constructed through systematic coarse-graining procedures, specifically via KDE. These high-dimensional fields were subsequently embedded into low-dimensional latent spaces using POD, allowing the dominant collective structures of the crowd dynamics to be captured in a compact representation. Within this reduced space, the temporal evolution of the system was formulated as the identification of discrete-time evolution operators acting on the latent state variables.

A key contribution of this work lies in showing that macroscopic crowd dynamics can be effectively modeled through operator learning in reduced latent spaces. Rather than attempting to identify explicit continuum equations, the proposed framework learns the effective solution operator governing the evolution of macroscopic observables. Both linear and nonlinear ROMs were considered for this purpose, including MVAR models and recurrent architectures based on LSTMs. Numerical experiments on benchmark crowd scenarios generated from the SFM showed that both linear MVAR and nonlinear LSTM models operating in carefully constructed latent spaces can achieve accurate long-horizon predictions while maintaining computational efficiency and interpretability. In particular, these models demonstrated substantial speed improvements relative to

conventional microscopic simulation while preserving key physical properties such as mass conservation by design through the POD reconstruction.

From a conceptual viewpoint, the results support a shift from traditional equation-based modeling toward variants of the EF paradigm for complex multi-agent systems. In this setting, the focus is placed not on deriving closed-form macroscopic equations but on identifying effective surrogate reduced-order models that map macroscopic states reliably forward in time. This perspective enables predictive modeling of emergent collective dynamics even when explicit governing equations are unknown or intractable, thereby expanding the range of systems that can be analyzed through principled reduced-order representations.

Beyond predictive modeling, the thesis extended the data-driven modeling multiscale framework to the problem of macroscopic state estimation. A PIML-based methodology was introduced for constructing nonlinear discrete-time observers through the identification of transformation operators that map nonlinear system dynamics into linear observer coordinates. The observer design problem was formulated as the solution of a system of functional equations defining this transformation, which was approximated using PINNs in particular. Numerical experiments on benchmark dynamical systems demonstrated that the proposed approach can achieve accurate state reconstruction.

Importantly, the observer framework was successfully deployed on the crowd dynamics problem by exploiting particularly the LSTM-based ROM, which can be interpreted as a discrete nonlinear system and enables the identification of the coordinate transformation $T(\mathbf{x})$. In this formulation, the observer operates autonomously in the transformed coordinates and does not require direct access to the original data-driven solution operator during closed-loop simulations, highlighting the structural role of the learned transformation operator as a bridge between nonlinear latent dynamics and linear estimation mechanisms.

In addition to estimation, the thesis explored the extension of the proposed framework to control-oriented formulations for complex dynamical systems. In particular, a PIML methodology was developed to construct nonlinear control transformations for discrete nonlinear systems, making the approach naturally compatible with the LSTM-based ROM. The proposed formulation enables the design of stabilization and regulation strategies within a single-step framework that remains consistent with the data-driven modeling paradigm adopted throughout the thesis. While the control methodology was validated on benchmark dynamical systems, its full integration with the learned macroscopic crowd-dynamics pipeline represents an important direction for future research. The primary contribution of this thesis in this direction is the establishment of the theoretical and computational foundations required for the control of learned reduced-order nonlinear systems within a data-driven framework. In this sense, the methodological components necessary for such an integration are already in place. The remaining effort lies primarily on the application side, namely in the formulation of appropriate crowd-management objectives and the incorporation of physically meaningful control actions

within the learned macroscopic dynamics. Addressing these aspects would enable the deployment and validation of the proposed control architecture in realistic crowd-dynamics scenarios, thereby completing the modeling–estimation–control framework developed throughout this thesis.

At this point, it is worth noting that there are still several limitations and interesting open challenges. First, the reliability and generalization properties of the learned latent representations depend on the quality and spatio-temporal sparsity of the training data. Significant departures from the configurations represented in the observation set may lead to reduced accuracy and generalization of the learned models.

Moreover, a central limitation of the proposed framework arises from its reliance on linear reduced-order representations constructed via POD. As is well known, POD-based models are strongly dependent on the geometry of the domain and the boundary conditions represented in the training data, this limitation is inherited by the present methodology: since the reduced latent space is constructed from a finite ensemble of observations, the resulting linear subspace may fail to capture the underlying solution manifold, leading to a degradation in predictive performance.

Addressing these limitations motivates several research directions. In particular, neural operator learning methods, such as Fourier Neural Operators and Deep Operator Networks, provide a promising alternative, as they are designed to learn mappings between function spaces and can generalize, in principle, across families of geometries and boundary conditions. Integrating such approaches within the present framework would enable more flexible and geometry-agnostic macroscopic models.

Another important aspect concerns the distinction between mass conservation and element-wise positivity of the reconstructed density fields. While the proposed framework guarantees mass conservation by construction through the POD-based reconstruction, it does not enforce non-negativity at the pointwise level. This limitation, as discussed before, is not specific to the present methodology but is a general property of linear unconstrained reconstruction techniques, including POD.

Even though the total mass of the reconstructed density field is preserved, small negative values may arise in regions where density approaches zero, as a consequence of the linear superposition of modes. In numerical experiments considered in this thesis, such deviations remain negligible and do not affect the overall macroscopic behavior. Nevertheless, ensuring elementwise positivity represents an additional constraint that is not enforced in the present formulation. Addressing this limitation would require the use of constrained reconstruction techniques or nonlinear latent representations. For instance, approaches based on nearest-neighbor (kNN) reconstruction in nonlinear manifold learning spaces can guarantee non-negativity by construction, albeit at an increased computational cost.

More generally, nonlinear manifold learning techniques such as diffusion maps and

autoencoder-based embeddings provide a more flexible framework for representing macroscopic states. In particular, diffusion maps exploit the intrinsic geometry of the data by constructing embeddings based on diffusion distances, which are robust to noise and capable of capturing nonlinear relationships between system states. This makes them particularly suitable for representing dynamics evolving on curved manifolds that cannot be accurately approximated by linear subspaces. Incorporating such techniques within the proposed framework constitutes a promising direction for future research.

Furthermore, recent developments in data-driven modeling provide complementary approaches that could enhance the proposed framework. Transformer-based architectures have demonstrated strong performance in long-horizon prediction tasks, while latent Neural Ordinary Differential Equations offer a continuous-time formulation for learning reduced dynamics. Similarly, Learning Effective Dynamics (LED) frameworks emphasize structure preservation and interpretability. A systematic comparison between these approaches and the operator-based methodology developed in this thesis would provide valuable insight into their respective advantages and limitations.

Another important direction concerns the incorporation of uncertainty quantification into the macroscopic modeling framework. In practical applications, uncertainties arise from measurement noise, variability in initial conditions, and modeling approximations. While the present work focused on deterministic predictions, probabilistic extensions could be naturally incorporated, particularly within the linear MVAR framework, whose Gaussian structure allows for interpretable uncertainty propagation through the latent dynamics.

Beyond uncertainty quantification, an additional question concerns the influence of the parameters defining the underlying microscopic dynamics. In the numerical experiments presented in this thesis, the SFM parameters were fixed according to commonly adopted values in the literature [15, 117, 119], allowing the analysis to focus on the identification of macroscopic dynamics and the assessment of generalization across unseen initial conditions. While the proposed framework demonstrated robustness with respect to perturbations of the initial conditions of the SFM, as discussed in Chapter 5, a systematic study of its sensitivity to variations in microscopic model parameters was beyond the scope of the present work. Such an analysis would be particularly relevant when considering heterogeneous pedestrian populations or parameter-dependent crowd behaviors. Extending the methodology toward parametric reduced-order representations and parameter-aware operator learning therefore constitutes an interesting direction for future research.

Finally, extending the framework to real-world trajectory data represents a crucial step toward practical deployment. Although this work focused on synthetic data generated from agent-based simulations, the proposed methodology is not tied to a particular microscopic model and is, in principle, applicable to alternative agent-based descriptions as well as empirical trajectory datasets. Such extensions would enable validation under realistic conditions and support applications in crowd management, urban mobility, and

safety-critical scenarios. Naturally, different microscopic models and real-world datasets may exhibit varying levels of noise, sampling resolution, measurement uncertainty, and data incompleteness, which can affect the quality of the resulting reduced-order representations and learned operators. Addressing these issues may require additional preprocessing, filtering, or data-curation procedures prior to the application of the proposed pipeline. However, these challenges are inherent to data-driven modeling in general and do not constitute limitations specific to the framework developed in this thesis.

Overall, this thesis demonstrates that the proposed data-driven framework provides a viable and flexible approach for connecting microscopic simulations with macroscopic prediction, estimation, and potentially control. By combining coarse-graining, reduced-order modeling, operator learning, and physics-informed methods, it contributes to a unified computational paradigm for understanding and governing emergent collective dynamics.

8.1 List of publications

1. H. V. Alvarez, D. G. Patsatzis, L. Russo, I. G. Kevrekidis, and C. Siettos, "Next generation equation-free multiscale modelling of crowd dynamics via machine learning," *Journal of Computational Physics*, vol. 559, 114881, 2026.
DOI: 10.1016/j.jcp.2026.114881

2. H. V. Alvarez, G. Fabiani, N. Kazantzis, C. Siettos, and I. G. Kevrekidis, "Nonlinear discrete-time observers with physics-informed neural networks," *Chaos, Solitons & Fractals*, vol. 168, p. 112697, 2024.
DOI: 10.1016/j.chaos.2024.112697

3. H. V. Alvarez, G. Fabiani, N. Kazantzis, C. Siettos, and I. G. Kevrekidis, "Discrete-time nonlinear feedback linearization via physics-informed machine learning," *Journal of Computational Physics*, vol. 492, p. 112408, 2023.
DOI: 10.1016/j.jcp.2023.112408

Appendix A

Density fields extraction via Kernel Density Estimation

Abstract Here, we provide additional details for the first step of the proposed framework, regarding the extraction of continuous density profiles from discrete pedestrians' positions. As already discussed in Chapter 3, we used KDE for this task, following the general form in Eq. (3.25); here, we elaborate on the kernel selection, bandwidth tuning, and adjustments for boundaries and obstacles.

The choice of kernel function directly influences the smoothness and continuity of the density estimate. Common options include the uniform, triangular, Epanechnikov, and Gaussian kernels [165, 204]. The Epanechnikov kernel is theoretically optimal for minimizing mean integrated squared error [205], but its bounded support can limit flexibility in practice. In contrast, the Gaussian kernel is widely preferred due to its smoothness, infinite support, and analytical tractability. Its multivariate form is given by:

$$K_h(\mathbf{x} - \mathbf{x}_i(t)) = \frac{1}{2\pi \det(\mathbf{H})^{1/2}} \exp\left(-\tfrac{1}{2}(\mathbf{x} - \mathbf{x}_i(t))^\top \mathbf{H}^{-1}(\mathbf{x} - \boldsymbol{x}_i(t))\right), \tag{A.1}$$

where $\mathbf{H}$ is the symmetric positive-definite bandwidth matrix, $\mathbf{x} \in \mathbb{R}^2$ is the spatial domain and $\boldsymbol{x}_i(t) \in \mathbb{R}^2$ is the position of the i-th pedestrian.

The choice of the bandwidth matrix, which in this case takes the form $\mathbf{H} = \mathrm{diag}(h_x, h_y)$, plays a central role in the KDE estimate. Small values of h_x/h_y result in undersmoothing and high variance, while large values can oversmooth the density and obscure relevant features. Silverman's rule of thumb [165], originally formulated for univariate distributions, extends naturally to multivariate settings, indicating an optimal bandwidth:

$$h_i = \sigma_i \left(\frac{4}{N(d+2)}\right)^{1/(d+4)}, \tag{A.2}$$

where N is the number of data points, d is the number of spatial dimensions, in this case $d = 2$, and σ_i is the standard deviation of the pedestrian positions along $i = x, y$ spatial directions. However, in our case, Silverman's rule resulted in wide bandwidths, which did not allow us to consider sufficient detail in the density profiles. We thus tuned h_x/h_y to the values reported in Chapter 3.

The standard symmetric kernels are known to introduce bias near domain boundaries, when handling cases where *periodic boundary conditions* are imposed, such as in the x-direction of pedestrian flow in our configuration. These artifacts arise because the kernel's support extends beyond the domain, failing to reflect the periodic structure of the system. To mitigate these artifacts, we define an extended support for the KDE in the x-direction. Let the physical domain boundaries be $x = a$ and $x = b$, and introduce a small displacement $\Delta x = \frac{b-a}{n}$, where n defines the extension of the domain along x-direction; here, we tuned $n = 5$. We then define the extended domain as:

$$\Omega^{\text{ext}} = [a - \Delta x, b + \Delta x] \times [c, d], \tag{A.3}$$

while the support in the y-direction remains unbounded. To enforce continuity and reduce boundary-induced artifacts in the x-direction, we further augment the observed positions with "ghost" particles near the domain edges. Let $\epsilon = \Delta x$ define a threshold for proximity to the boundary. For each position $\boldsymbol{x}_i(t) = (x_i(t), y_i(t))$ such that $x_i(t) \leq a + \epsilon$, we introduce a "mirrored" pedestrians at

$$\boldsymbol{x}_i^{\text{aug}1}(t) = (x_i(t) - (b - a), y_i(t)).$$

Similarly, for every $\boldsymbol{x}_i(t)$ such that $x_i(t) \geq b - \epsilon$, we introduce a "mirrored" pedestrians at

$$\boldsymbol{x}_i^{\text{aug2}}(t) = (x_i(t) + (b - a), y_i(t)).$$

The resulting augmented set of pedestrian positions used for KDE is then given by

$$\{\boldsymbol{x}_i'(t)\} = \{\boldsymbol{x}_i(t)\} \cup \{\boldsymbol{x}_i^{\text{aug1}}(t)\} \cup \{\boldsymbol{x}_i^{\text{aug2}}(t)\}. \tag{A.4}$$

A pseudo-code for the estimation of the density in $\Omega \subset \mathbb{R}^2$ is given in Algorithm 1. In

Algorithm 1 Kernel Density Estimation (KDE) for extracting density fields from pedestrian positions.

Require: Number of pedestrians N, computational domain $\Omega = [a, b] \times [c, d]$, spatial discretization n_x/n_y, obstacle subdomain $\Gamma = [a_o, b_o] \times [c_o, d_o] \subset \Omega$ and pedestrian positions $\{\mathbf{x}_i(t)\}_{i=1}^N \in \Omega \backslash \Gamma$

Set bandwidth $\mathbf{H} = \text{diag}(h_x, h_y)$.
Set augmentation on x-direction $\Delta x = (b - a)/n$
Define extended support Ω^{ext} to handle periodic boundary conditions ▷ *Use Eq. (A.3)*
Discretize extended domain Ω^{ext} in $(n_x + 2n_x/n) \times n_y$ cells
Introduce "mirrored" pedestrians to form $\{\mathbf{x}_i'(t)\}$ ▷ *Use Eq. (A.4)*
Initialize zero density field $\rho((x_i, y_j), t)$ across the extended domain Ω^{ext}
for each pedestrian $p = 1, 2, \ldots, N$ **do** ▷ *Account for "mirrored" pedestrians*
    Compute kernel contribution $K_h(\mathbf{x} - \mathbf{x}_p(t))$ over the extended support Ω^{ext} ▷ *Use Eq. (A.1)*
    Add contribution to density field $\rho((x_i, y_j), t)$
end for
Normalize $\rho((x_i, y_j), t)$ with total number of pedestrians, N and "mirrored" ones
Retrieve density $\rho((x_i, y_j), t)$ in computational domain Ω
Employ binary mask function to ensure zero density in obstacle subdomain Γ, by setting ▷ *See Eq. (A.5)*
$\rho((x_i, y_i), t) \leftarrow \mathcal{M}(x_i, y_i) \cdot \rho((x_i, y_i), t)$
Compute total estimated density over the entire domain $S(t) = \sum_{i,j=1}^{n_x, n_y} \rho((x_i, y_j), t)\delta x \delta y$
Normalize density field $\rho((x_i, y_j), t) \leftarrow \rho((x_i, y_j), t)/S(t)$
return Normalized density $\rho((x_i, y_j), t)$ across computational domain Ω

the presence of obstacles, formally described as a subdomain $\Gamma \subset \Omega$, the standard KDE must be adjusted to ensure that the density vanishes within and along the boundaries of these regions. This is typically achieved by introducing a binary mask function $\mathcal{M}(x, y)$:

$$M(x, y) = \begin{cases} 0, & \text{if } (x, y) \in \Gamma \subset \Omega, \\ 1, & \text{otherwise.} \end{cases} \tag{A.5}$$

which is then applied to the initially estimated densities to enforce zero density in Γ, implying:

$$\rho((x_i, y_j), t) = \mathcal{M}(x_i, y_j)\, \rho((x_i, y_j), t), \quad \forall (x_i, y_j) \in \Omega. \tag{A.6}$$

Appendix B

Initial Conditions of Microscopic Distributions

Abstract Here, we describe the initial conditions considered in the SFM to construct the datasets, upon which the proposed framework is trained and tested. We considered zero initial velocities for all pedestrians, while we used the following distributions for initializing the position $\boldsymbol{x}_i = (x_i, y_i) \in \Omega$ of the i-th pedestrian.

Positions are sampled from:

- Uniform distributions over the domain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \subset \Omega$:

$$p(x_i, y_i) = \mathcal{U}\left([x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]\right). \tag{B.1}$$

- Two dimensional normal distributions with means (μ_x, μ_y) and standard deviations (σ_x, σ_y):

$$p(x_i, y_i) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{(x_i - \mu_x)^2}{2\sigma_x^2} - \frac{(y_i - \mu_y)^2}{2\sigma_y^2}\right), \tag{B.2}$$

- "Double bell-shaped" independent joint distributions, which are comprised by normal distributions along the y–axis with mean μ_y and standard deviation σ_y, and a mixture of normal distributions along the x–axis with means (μ_{x_1}, μ_{x_2}) and

standard deviations $(\sigma_{x_1}, \sigma_{x_2})$:

$$p(x_i, y_i) = \frac{1}{\sqrt{2\pi}\,\sigma_y} \exp\left(-\frac{(y-\mu_y)^2}{2\sigma_y^2}\right) \times \frac{1}{2}\left[\frac{1}{\sqrt{2\pi}\,\sigma_{x_1}} \exp\left(-\frac{(x-\mu_{x_1})^2}{2\sigma_{x_1}^2}\right) + \frac{1}{\sqrt{2\pi}\,\sigma_{x_2}} \exp\left(-\frac{(x-\mu_{x_2})^2}{2\sigma_{x_2}^2}\right)\right]. \tag{B.3}$$

- Cosine distributions centered at (c_x, c_y) with scale parameters (s_x, s_y):

$$p(x_i, y_i) = \frac{1}{\pi s_x s_y} \cos\left(\frac{x_i - c_x}{s_x}\right) \cos\left(\frac{y_i - c_y}{s_y}\right). \tag{B.4}$$

The training and testing data sets were generated from $C = 20$ initial conditions, with 5 samples drawn from each of the four distributions described above. To ensure plausible and varied crowd configurations, the parameters for these distributions were selected heuristically to provide a broad spatial coverage while mitigating boundary artifacts. Furthermore, during the sampling process, pedestrian positions were resampled if they were within a critical distance equal to $2r_i$ of an existing pedestrian or a wall/obstacle to prevent initial overlaps and collisions. The specific parameter configurations used for the training and testing data sets are detailed Table B.1 and Table B.2, respectively.

Table B.1: Microscopic initial distribution configurations and parameters for constructing the training dataset.

Case c	**Distribution Type**	**Parameters**
1	Uniform, Eq. (B.1)	$x_{\min} = 2, x_{\max} = 15, y_{\min} = 3, y_{\max} = 9$
2	Uniform, Eq. (B.1)	$x_{\min} = 33, x_{\max} = 46, y_{\min} = 3, y_{\max} = 9$
3	Uniform, Eq. (B.1)	$x_{\min} = 2, x_{\max} = 20, y_{\min} = 4, y_{\max} = 8$
4	Uniform, Eq. (B.1)	$x_{\min} = 28, x_{\max} = 46, y_{\max} = 4, y_{\max} = 8$
5	Uniform, Eq. (B.1)	$x_{\min} = 3, x_{\max} = 17, y_{\max} = 4, y_{\max} = 8$
6	Gaussian, Eq. (B.2)	$\mu_x = 10,\ \mu_y = 6, \sigma_x = 1.5,\ \sigma_y = 1.5$
7	Gaussian, Eq. (B.2)	$\mu_x = 10,\ \mu_y = 6, \sigma_x = 2,\ \sigma_y = 1.5$
8	Gaussian, Eq. (B.2)	$\mu_x = 38,\ \mu_y = 6, \sigma_x = 2,\ \sigma_y = 1.5$
9	Gaussian, Eq. (B.2)	$\mu_x = 38,\ \mu_y = 6, \sigma_x = 1.5,\ \sigma_y = 1.5$
10	Gaussian, Eq. (B.2)	$\mu_x = 37,\ \mu_y = 5, \sigma_x = 1.5,\ \sigma_y = 1.5$
11	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 8,\ \mu_y = 7, \sigma_{x_1} = 1.2,\ \sigma_y = 1.5,$ $\mu_{x_2} = 12,\ \sigma_{x_2} = 0.8$
12	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 10,\ \mu_y = 6, \sigma_{x_1} = 1.2,\ \sigma_y = 1.5,$ $\mu_{x_2} = 14,\ \sigma_{x_2} = 1$
13	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 28,\ \mu_y = 6, \sigma_{x_1} = 1.3,\ \sigma_y = 1.5,$ $\mu_{x_2} = 32,\ \sigma_{x_2} = 1.1$
14	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 36,\ \mu_y = 6, \sigma_{x_1} = 1.4,\ \sigma_y = 1.5,$ $\mu_{x_2} = 40,\ \sigma_{x_2} = 1.2$
15	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 9,\ \mu_y = 6, \sigma_{x_1} = 0.8,\ \sigma_y = 1.5,$ $\mu_{x_2} = 11,\ \sigma_{x_2} = 1$
16	Cosine, Eq. (B.4)	$c_x = 10,\ c_y = 6, s_x = 10,\ s_y = 3$
17	Cosine, Eq. (B.4)	$c_x = 20,\ c_y = 8, s_x = 10,\ s_y = 3$
18	Cosine, Eq. (B.4)	$c_x = 30,\ c_y = 7, s_x = 10,\ s_y = 3$
19	Cosine, Eq. (B.4)	$c_x = 40,\ c_y = 6, s_x = 10,\ s_y = 3$
20	Cosine, Eq. (B.4)	$c_x = 10,\ c_y = 6, s_x = 9,\ s_y = 3$

Table B.2: Microscopic initial distribution configurations and parameters for constructing the testing dataset.

Case c	**Distribution Type**	**Parameters**
1	Uniform, Eq. (B.1)	$x_{\min} = 2,\ x_{\max} = 16,\ y_{\min} = 3,\ y_{\max} = 10$
2	Uniform, Eq. (B.1)	$x_{\min} = 32,\ x_{\max} = 46,\ y_{\min} = 3,\ y_{\max} = 10$
3	Uniform, Eq. (B.1)	$x_{\min} = 2,\ x_{\max} = 19,\ y_{\min} = 4,\ y_{\max} = 8$
4	Uniform, Eq. (B.1)	$x_{\min} = 29,\ x_{\max} = 46,\ y_{\min} = 4,\ y_{\max} = 8$
5	Uniform, Eq. (B.1)	$x_{\min} = 4,\ x_{\max} = 17,\ y_{\min} = 4,\ y_{\max} = 8$
6	Gaussian, Eq. (B.2)	$\mu_x = 12,\ \mu_y = 7,\ \sigma_x = 2,\ \sigma_y = 1.5$
7	Gaussian, Eq. (B.2)	$\mu_x = 11,\ \mu_y = 5,\ \sigma_x = 1.5,\ \sigma_y = 2$
8	Gaussian, Eq. (B.2)	$\mu_x = 38,\ \mu_y = 6,\ \sigma_x = 2,\ \sigma_y = 1.5$
9	Gaussian, Eq. (B.2)	$\mu_x = 39,\ \mu_y = 5,\ \sigma_x = 1.5,\ \sigma_y = 1.5$
10	Gaussian, Eq. (B.2)	$\mu_x = 37,\ \mu_y = 7,\ \sigma_x = 1.5,\ \sigma_y = 1.5$
11	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 7,\ \mu_y = 7,\ \sigma_{x_1} = 1.1,\ \sigma_y = 1.5,$ $\mu_{x_2} = 11,\ \sigma_{x_2} = 0.9$
12	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 10,\ \mu_y = 7,\ \sigma_{x_1} = 1.2,\ \sigma_y = 1.5,$ $\mu_{x_2} = 14,\ \sigma_{x_2} = 1$
13	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 29,\ \mu_y = 6,\ \sigma_{x_1} = 1.3,\ \sigma_y = 1.5,$ $\mu_{x_2} = 33,\ \sigma_{x_2} = 1.1$
14	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 37,\ \mu_y = 6,\ \sigma_{x_1} = 1.4,\ \sigma_y = 1.5,$ $\mu_{x_2} = 41,\ \sigma_{x_2} = 1.2$
15	Double Gaussian, Eq. (B.3)	$\mu_{x_1} = 10,\ \mu_y = 6,\ \sigma_{x_1} = 0.8,\ \sigma_y = 1.5,$ $\mu_{x_2} = 12,\ \sigma_{x_2} = 1$
16	Cosine, Eq. (B.4)	$c_x = 15,\ c_y = 4,\ s_x = 8,\ s_y = 3$
17	Cosine, Eq. (B.4)	$c_x = 25,\ c_y = 6,\ s_x = 10,\ s_y = 3$
18	Cosine, Eq. (B.4)	$c_x = 35,\ c_y = 5,\ s_x = 10,\ s_y = 3$
19	Cosine, Eq. (B.4)	$c_x = 45,\ c_y = 7,\ s_x = 5,\ s_y = 3$
20	Cosine, Eq. (B.4)	$c_x = 15,\ c_y = 6,\ s_x = 7,\ s_y = 3$

Appendix C

Open-loop prediction errors and closed-loop reconstructions for the unidirectional flow case

Abstract Here, we provide the detailed open-loop prediction errors and the closed-loop reconstructions of the proposed framework for the unidirectional flow case for models MVAR(4), LSTM(4) and LSTM(9) discussed in Section 5.5.1.

The open-loop setting prediction of the proposed restrict-evolve with ROM-lift framework in Eq. (5.3) reads:

$$\hat{x}^{(c)}(t_k) = \mathcal{L}\left(\hat{\boldsymbol{y}}^{(c)}(t_k)\right), \qquad \hat{\boldsymbol{y}}^{(c)}(t_k) = \Phi_{\text{ROM}}\left(\boldsymbol{y}^{(c)}(t_{k-1}), \ldots, \boldsymbol{y}^{(c)}(t_{k-w}); \mathbf{p}\right), \tag{C.1}$$

where $\boldsymbol{y}^{(c)}(t_{k-j}) = \mathcal{R}(x^{(c)}(t_{k-j}))$ are restricted embeddings of observed densities for $j = 1, \ldots, w$. Similarly to the close-loop predictions in Eq. (5.19), the open-loop prediction expression is used in Eq. (5.4.6) to quantify the one-step prediction accuracy of the proposed framework.

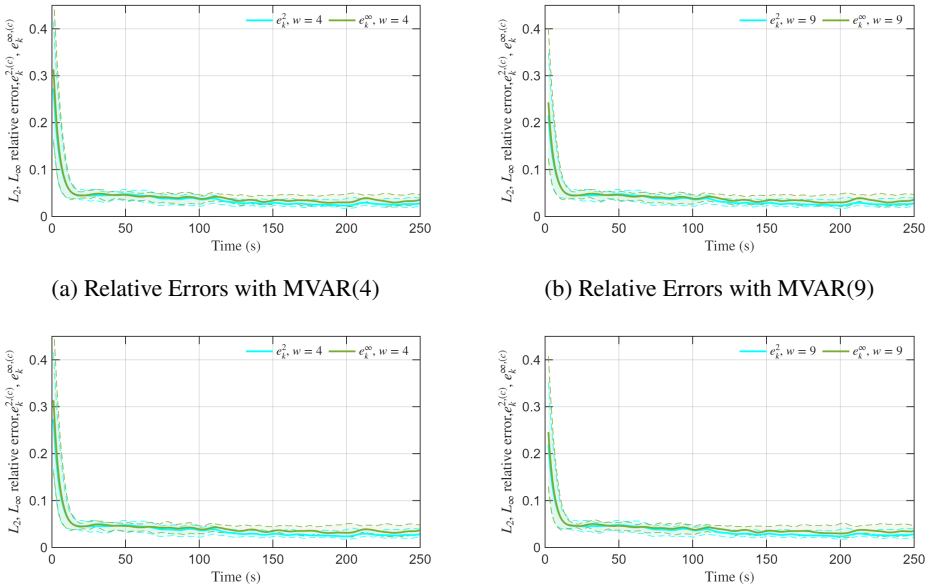
Table C.1 provides a summary of the relative reconstructed errors in the ambient space, for one-step predictions, for all four ROMs, including the mean error and the 10–90% error percentiles, aggregated *over all cases and time steps* in the testing set.

Table C.1: Open-loop (one-step) prediction errors in the ambient–density profile–space, for the testing set using the trained MVAR and LSTM (best out of 50 training runs) models for the unidirectional flow case. The relative reconstructed L_1, L_2 and L_∞ errors $e_k^{1,(c)}$, $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ in Eq. (5.4.6) are reported for the MVAR(4), MVAR(9), LSTM(4), and LSTM(9) latent dynamics models. Mean and 10–90% error percentiles are shown *over all the* $C = 20$ cases and $K = \{991, 996\}$ time steps (for width $w = \{9, 4\}$, respectively).

Model	L_1 error, $e_k^{1,(c)}$	L_2 error, $e_k^{2,(c)}$	L_∞ error, $e_k^{\infty,(c)}$
MVAR(4)	0.048 (0.033, 0.065)	0.038 (0.027, 0.051)	0.042 (0.030, 0.057)
MVAR(9)	0.047 (0.032, 0.063)	0.037 (0.026, 0.049)	0.041 (0.030, 0.055)
LSTM(4)	0.048 (0.034, 0.065)	0.038 (0.027, 0.051)	0.043 (0.032, 0.058)
LSTM(9)	0.047 (0.033, 0.063)	0.037 (0.026, 0.049)	0.042 (0.031, 0.056)

Figure C.1 displays the relative reconstruction errors $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ and their 10–90% percentile ranges for one-step predictions via the open-loop time-stepper (Eq. (C.1)). Errors are averaged over the $C = 20$ cases of different, unseen initial conditions considered, across the time steps $k = 1, \ldots, 1000$.

Figures C.2, C.3 and C.4 compare the ground truth normalized density fields $x^{(c)}(t_k)$ and the predicted ones $\hat{x}^{(c)}(t_k)$ for closed-loop predictions (Eq. (5.19)) using MVAR(4), LSTM(4), and LSTM(9) models, respectively. The test case uses a Gaussian initialization (6th row of Table B.2) at four time steps ($k = 250, 500, 780, 930$), during which the pedestrian crowd does not pass through the corridor boundary. Panels (a,c,e,g) show the ground truth, while panels (b,d,f,h) present the corresponding closed-loop predictions. The spatial distribution of the resulting absolute errors is shown in Figure C.5 at time steps $k = 250, 930$.



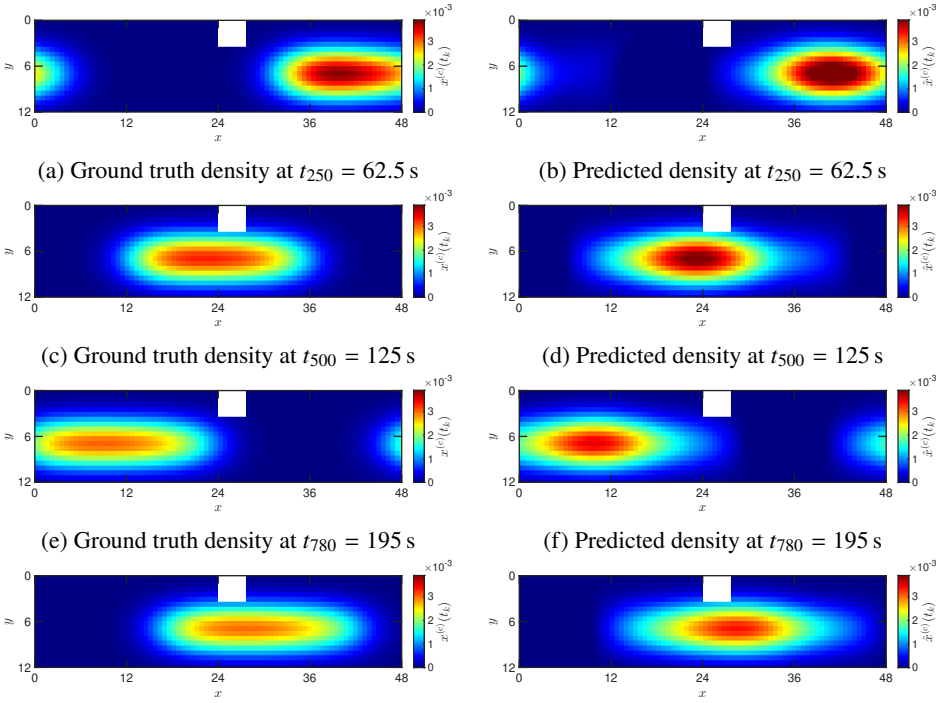
(a) Relative Errors with MVAR(4)

(b) Relative Errors with MVAR(9)

(c) Relative Errors with LSTM(4)

(d) Relative Errors with LSTM(9)

Figure C.1: One-step (open-loop) prediction errors, in the ambient–density profile–space, for the testing set over time, using trained MVAR and LSTM models for the unidirectional flow case. The relative reconstructed L_2 (cyan) and L_∞ (green) errors $e_k^{2,(c)}$ and $e_k^{\infty,(c)}$ Eq. (5.4.6) (using the one-step prediction in Eq. (C.1)) are shown for the MVAR(4) panel (a), MVAR(9) panel (b), LSTM(4) panel (c), and LSTM(9) panel (d) latent dynamics models. Mean relative errors (solid) and 10–90% error percentiles (dashed) are shown over $C = 20$ cases per time step. For the LSTM models, results correspond to the model achieving the best training performance out of the 50 independent runs.



(a) Ground truth density at $t_{250} = 62.5$ s

(b) Predicted density at $t_{250} = 62.5$ s

(c) Ground truth density at $t_{500} = 125$ s

(d) Predicted density at $t_{500} = 125$ s

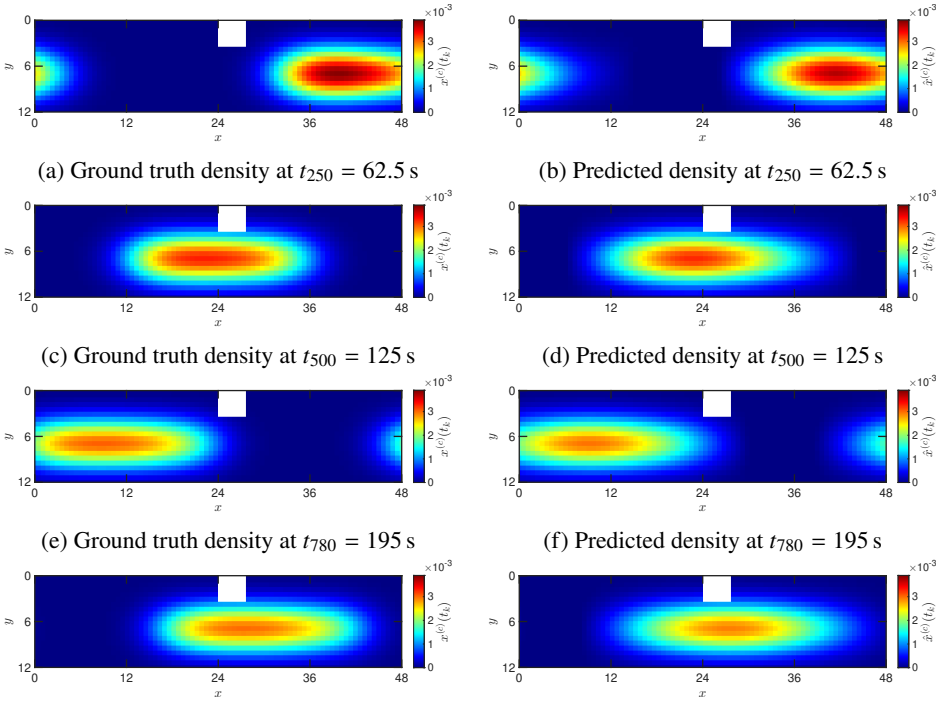
(e) Ground truth density at $t_{780} = 195$ s

(f) Predicted density at $t_{780} = 195$ s

(g) Ground truth density at $t_{930} = 232.5$ s

(h) Predicted density at $t_{930} = 232.5$ s

Figure C.2: Comparison of the "ground truth" $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ normalized densities via the closed-loop time-stepper in Eq. (5.19) for the unidirectional flow case, using the MVAR(4) model for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels (a), (c), (e), and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f), and (h) show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.



(a) Ground truth density at $t_{250} = 62.5$ s

(b) Predicted density at $t_{250} = 62.5$ s

(c) Ground truth density at $t_{500} = 125$ s

(d) Predicted density at $t_{500} = 125$ s

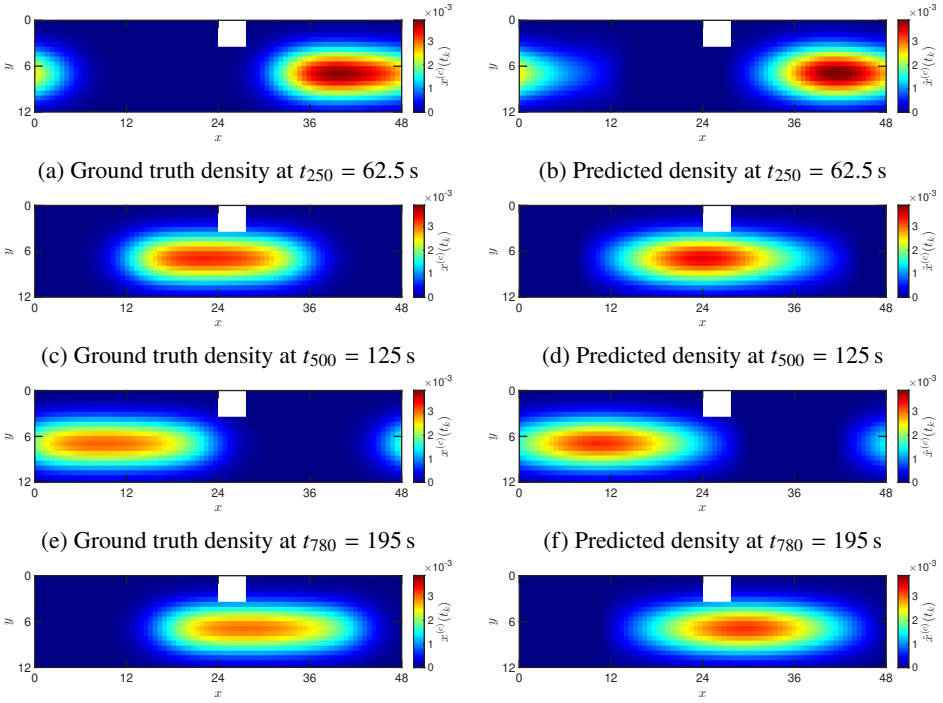
(e) Ground truth density at $t_{780} = 195$ s

(f) Predicted density at $t_{780} = 195$ s

(g) Ground truth density at $t_{930} = 232.5$ s

(h) Predicted density at $t_{930} = 232.5$ s

Figure C.3: Comparison of the "ground truth" $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ normalized densities via the closed-loop time-stepper in Eq. (5.19) for the unidirectional flow case, using the LSTM(4) model for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels (a), (c), (e), and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f), and (h), show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.



(a) Ground truth density at $t_{250} = 62.5$ s

(b) Predicted density at $t_{250} = 62.5$ s

(c) Ground truth density at $t_{500} = 125$ s

(d) Predicted density at $t_{500} = 125$ s

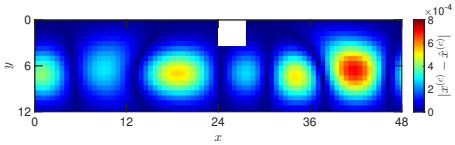
(e) Ground truth density at $t_{780} = 195$ s

(f) Predicted density at $t_{780} = 195$ s

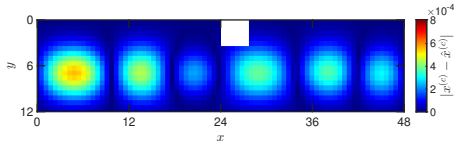
(g) Ground truth density at $t_{930} = 232.5$ s

(h) Predicted density at $t_{930} = 232.5$ s

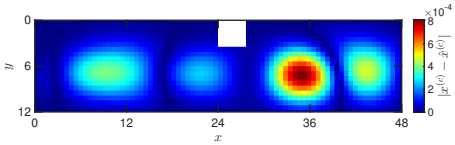
Figure C.4: Comparison of the "ground truth" $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ normalized densities via the closed-loop time-stepper in Eq. (5.19) for the unidirectional flow case, using the LSTM(9) model for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels (a), (c), (e), and (g) show the "ground truth" density distribution in Ω, while panels (b), (d), (f), and (h) show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.



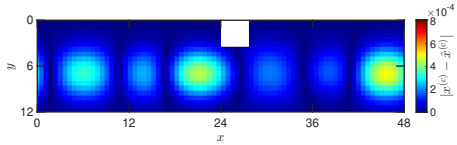
(a) MVAR(4) absolute error at time $t_{250} = 62.5$ s



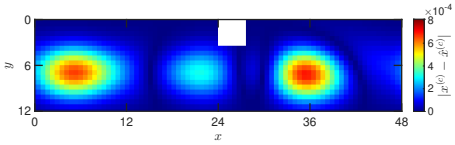
(b) MVAR(4) absolute error at time $t_{930} = 232.5$ s



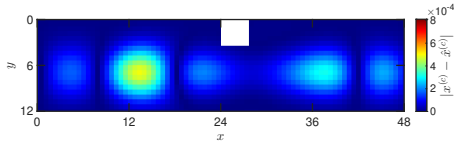
(c) MVAR(9) absolute error at time $t_{250} = 62.5$ s



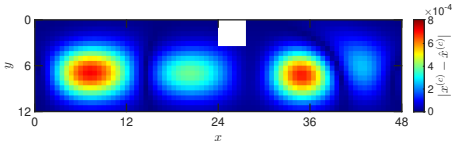
(d) MVAR(9) absolute error at time $t_{930} = 232.5$ s



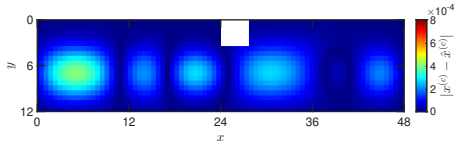
(e) LSTM(4) absolute error at time $t_{250} = 62.5$ s



(f) LSTM(4) absolute error at time $t_{930} = 232.5$ s



(g) LSTM(9) absolute error at time $t_{250} = 62.5$ s



(h) LSTM(9) absolute error at time $t_{930} = 232.5$ s

Figure C.5: Absolute error between the "ground truth" $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ normalized densities for the unidirectional flow case, corresponding to the predictions in Figs. C.2, 5.5, C.3, and C.4. Panels (a)–(h) show the spatial distribution of the absolute error $|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)|$ in Ω at two indicative time steps $t_{250} = 62.5$ s (left column) and $t_{930} = 232.5$ s (right column). Panels (a,b) correspond to MVAR(4), (c,d) to MVAR(9), (e,f) to LSTM(4), and (g,h) to LSTM(9) models.

Appendix D

Detailed results for the counter-flow case

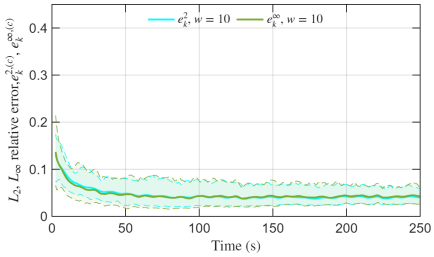
Abstract Here, we provide the detailed results of the proposed framework for the counterflow case discussed in Section 5.5.3, including baseline errors of the restriction and lifting operators, training of the MVAR and LSTM model, open-loop reconstruction performance.

Table D.1 summarizes the open-loop end-to-end performance of the proposed framework. Figure D.1 displays the relative reconstruction errors $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ per group and their 10–90% percentile ranges for one-step predictions via the open-loop time-stepper (Eq. (C.1)). Errors are averaged over the $C = 20$ cases of different, unseen initial conditions considered, across the time steps $k = 1, \ldots, 1000$.

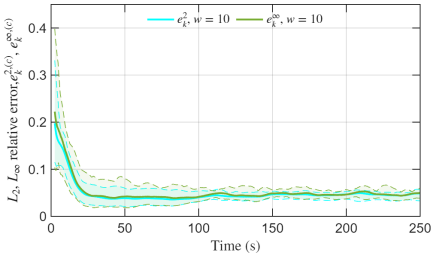
The test case uses a double gaussian initialization (15th row of Table B.2) at four time steps ($k = 250, 560, 780, 950$), during which the pedestrian groups do not pass through the corridor boundary. The two interacting populations are shown in red (group 1) and blue (group 2). Panels (a,c,e,g) show the ground truth, while panels (b,d,f,h) present the corresponding closed-loop predictions. The spatial distribution of the resulting absolute errors is shown in Figure D.2 at time steps $k = 250, 950$.

Table D.1: Open-loop (one-step) prediction errors in the ambient density profile space for the testing set using the trained MVAR and LSTM (best out of 50 training runs) models for the counterflow case. The relative reconstructed L_1, L_2, and L_∞ errors $e_k^{1,(c)}$, $e_k^{2,(c)}$, and $e_k^{\infty,(c)}$ in Eq. (5.4.6) are reported separately for group 1 and group 2 using the MVAR(10) and LSTM(10) latent dynamics models. Mean values and 10–90% error percentiles (in parentheses) are shown over all $C = 20$ cases and $K = 990$ time steps.

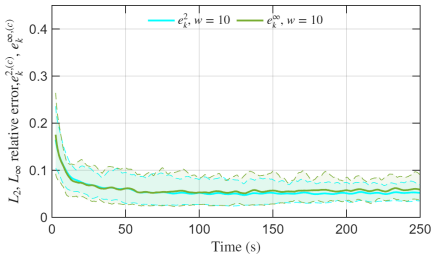
Model	L_1 error	L_2 error	L_∞ error
group 1			
MVAR(10)	0.063 (0.041, 0.100)	0.046 (0.027, 0.076)	0.046 (0.024, 0.078)
LSTM(10)	0.075 (0.049, 0.109)	0.057 (0.034, 0.086)	0.060 (0.032, 0.095)
group 2			
MVAR(10)	0.070 (0.049, 0.089)	0.048 (0.034, 0.063)	0.051 (0.034, 0.073)
LSTM(10)	0.082 (0.061, 0.101)	0.056 (0.041, 0.071)	0.064 (0.043, 0.092)



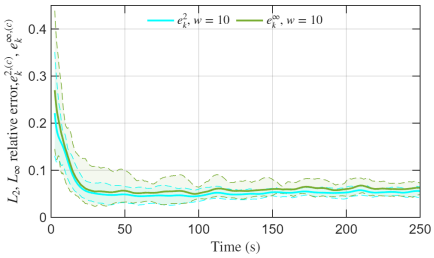
(a) Relative Errors with MVAR(10) (group 1)



(b) Relative Errors with MVAR(10) (group 2)

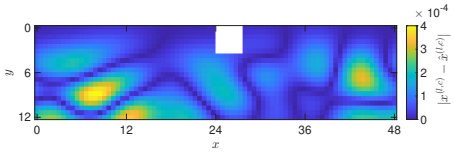


(c) Relative Errors with LSTM(10) (group 1)

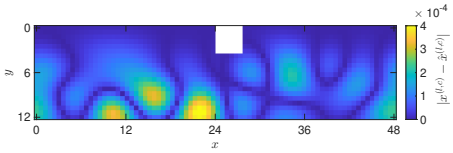


(d) Relative Errors with LSTM(10) (group 2)

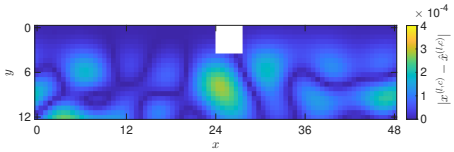
Figure D.1: One-step (open-loop) prediction errors, in the ambient—density profile—space, for the testing set over time, using trained MVAR and LSTM models for the counterflow case. The relative reconstructed L_2 (cyan) and L_∞ (green) errors $e_k^{2,(c)}$ and $e_k^{\infty,(c)}$ Eq. (5.4.6) (using the one-step prediction in Eq. (C.1)) are shown for the MVAR(10) and LSTM(10) latent dynamics models in panels (a,b) (c,d) per group, respectively. Mean relative errors (solid) and 10–90% error percentiles (dashed) are shown over $C = 20$ cases per time step. For the LSTM models, results correspond to the model achieving the best training performance out of the 50 independent runs.



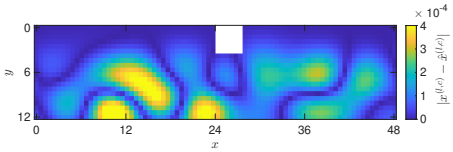
(a) MVAR(10) absolute error at time $t_{250} = 62.5\,\mathrm{s}$



(b) MVAR(10) absolute error at time $t_{950} = 237.5\,\mathrm{s}$



(c) LSTM(10) absolute error at time $t_{250} = 62.5\,\mathrm{s}$



(d) LSTM(10) absolute error at time $t_{950} = 237.5\,\mathrm{s}$

Figure D.2: Absolute error between the "ground truth" $x^{(l,c)}(t_k)$ and the predicted $\hat{x}^{(l,c)}(t_k)$ normalized densities in the counterflow case, corresponding to the predictions for groups $l = 1, 2$ in Figs. 5.8 and 5.9. Panels (a)–(d) show the spatial distribution of the absolute error $|x^{(l,c)}(t_k) - \hat{x}^{(l,c)}(t_k)|$ in Ω at two indicative time steps $t_{250} = 62.5\,\mathrm{s}$ (left column) and $t_{950} = 237.5\,\mathrm{s}$ (right column). Panels (a,b) correspond to MVAR(10), and panels (c,f) to LSTM(10) models.

Appendix E

Scientific Machine Learning

Abstract This appendix provides the foundational concepts underlying scientific machine learning, with particular emphasis on artificial neural networks and their principal variants. We review their mathematical formulation, universal approximation properties, and expressivity, as well as essential aspects of training, optimization, and generalization. Furthermore, we discuss key numerical considerations, including stability, conditioning, convergence behavior, and common sources of error that arise in practical implementations.

E.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a class of computational models inspired by biological nervous systems, in which collections of simple processing units (neurons) are interconnected to form a network capable of transforming inputs into outputs. Early conceptual foundations date back to the work of [206], who introduced a mathematical abstraction of a neuron based on threshold logic and demonstrated how networks of such units can implement logical computation. Building on this idea, the perceptron model [207] provided one of the first practical learning rules for adjusting connection weights from data, stimulating early interest in neural approaches. However, the representational limitations of single-layer perceptrons, most famously their inability to represent non-linearly separable functions, were emphasized by [208], contributing to a temporary decline in neural network research.

A major revival occurred in the 1980s with the (re)popularization of backpropagation for training multi-layer networks [209], enabling efficient gradient-based optimization of models with hidden layers. From a theoretical standpoint, this period and the years that followed produced foundational representational results. Universal approximation theorems established that feedforward neural networks with at least one hidden layer can approximate broad classes of continuous functions on compact domains, given sufficient width and appropriate nonlinearities [55, 210]. Further refinements related approximation rates to function regularity and model capacity [211], strengthening the view of ANNs as flexible, parameterized function approximators.

In parallel, neural networks became firmly embedded within the broader statistical learning framework [212], and advances in compute, data availability, and algorithmic design led to striking empirical successes. For instance, deep autoencoder models demonstrated effective nonlinear representation learning [213]. In computer vision, large-scale convolutional architectures achieved breakthrough performance on image recognition benchmarks [214], while in reinforcement learning deep networks enabled end-to-end control directly from high-dimensional observations [215]. In natural language processing, attention-based architectures and the Transformer model became a cornerstone for sequence modeling and large-scale language systems [216].

Deep Neural Networks (DNNs) are ANNs characterized by multiple hidden layers between input and output. Depth enables successive feature transformations, often supporting hierarchical representations in which earlier layers capture simple patterns and later layers combine them into more abstract concepts [217]. While this hierarchical structure can improve sample efficiency and predictive accuracy in many domains, increased depth and capacity also introduce substantial theoretical and practical challenges. In particular, training deep models corresponds to solving large-scale, highly nonconvex optimization problems, for which global optimality guarantees are generally unavailable. As a consequence, learning dynamics can be dominated by saddle points and flat regions of the loss landscape [218], sensitivity to initialization and hyperparameters, and instability due to vanishing or exploding gradients [57, 219]. These issues can lead to slow convergence,

suboptimal solutions, and significant variability across training runs, even when models share the same architecture and dataset.

From a representational perspective, classical universal approximation results guarantee that sufficiently wide feedforward neural networks can approximate arbitrary continuous functions on compact domains to arbitrary accuracy [55, 210]. However, these theorems are existential in nature: they do not provide constructive bounds on the number of parameters required, do not ensure that gradient-based training will find such approximating solutions, and do not address generalization beyond the training data. In practice, the gap between theoretical expressivity and effective learnability remains a central concern, motivating ongoing research into optimization algorithms, regularization strategies, architectural design principles, and constrained learning formulations. These considerations set the stage for the formal treatment of feedforward architectures and approximation properties in the next sections, as well as the training methodologies and physics-informed extensions discussed later in this chapter.

E.1.1 Feedforward Neural Networks

Feedforward neural networks (FNNs) are among the most basic, yet powerful and widely used artificial neural network architectures. They define a parametric mapping from an input space to an output space through a finite composition of affine transformations and componentwise nonlinear activation functions[55], with information propagating strictly from the input layer to the output layer.

Let $L \in \mathbb{N}$ denote the number of affine layers of the network. For $\ell = 0, \dots, L$, let $d_\ell \in \mathbb{N}$ denote the width of layer ℓ, with input dimension d_0 and output dimension d_L. Given an input vector $x \in \mathbb{R}^{d_0}$, the network output $\hat{y}(x;\theta) \in \mathbb{R}^{d_L}$ is defined recursively as follows. Set

$$h^{(0)} = x, \tag{E.1}$$

and for each layer $\ell = 1, \dots, L$, define

$$z^{(\ell)} = W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}, \tag{E.2}$$

where $W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is the weight matrix and $b^{(\ell)} \in \mathbb{R}^{d_\ell}$ is the bias vector. The hidden-layer activations are then given by

$$h^{(\ell)} = \phi^{(\ell)}\left(z^{(\ell)}\right), \qquad \ell = 1, \dots, L-1, \tag{E.3}$$

where each $\phi^{(\ell)} : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}$ acts componentwise through a scalar activation function, again denoted by $\phi^{(\ell)}$ by abuse of notation. The output layer is defined by

$$\hat{y}(x;\theta) = h^{(L)} = \phi^{(L)}\left(z^{(L)}\right). \tag{E.4}$$

The collection of trainable parameters is denoted by

$$\theta = \left\{\left(W^{(\ell)}, b^{(\ell)}\right)\right\}_{\ell=1}^{L}. \tag{E.5}$$

In many regression settings, the output activation $\phi^{(L)}$ is chosen as the identity map, whereas in classification tasks it may be taken as a softmax or logistic map, depending on the problem.

It is convenient to introduce the affine operators

$$T^{(\ell)}(u) = W^{(\ell)}u + b^{(\ell)}, \qquad \ell = 1, \dots, L. \tag{E.6}$$

With this notation, the network realizes the mapping

$$\hat{y}(x;\theta) = \left(\phi^{(L)} \circ T^{(L)} \circ \phi^{(L-1)} \circ T^{(L-1)} \circ \cdots \circ \phi^{(1)} \circ T^{(1)}\right)(x). \tag{E.7}$$

Thus, an FNN may be viewed as a finite-dimensional parametric function class generated by repeated composition of affine maps and nonlinearities. This representation makes explicit that the expressive power of the network is determined jointly by its depth, width, activation functions, and trainable parameters[220, 177, 212].

E.1.2 Universal Approximation

A fundamental theoretical property supporting the use of neural networks as function approximators is the *universal approximation theorem*. Informally, this result states that sufficiently large feedforward neural networks are capable of approximating broad classes of functions to arbitrary accuracy.

Early work on functional representation of multivariate functions can be traced back to Kolmogorov's superposition theorem. In the context of neural networks, the first rigorous approximation results were established in the late 1980s. Cybenko [55] proved that feedforward neural networks with a single hidden layer and sigmoidal activation functions are dense in the space of continuous functions on compact sets. Subsequent work by Hornik [56, 210] extended these results to broader classes of activation functions and network architectures. Later, Leshno et al. [221] established that a necessary and sufficient condition for universal approximation is that the activation function be non-polynomial.

We state a commonly used form of the universal approximation theorem.

Theorem E.1.1 (Universal Approximation)**.** Let $K \subset \mathbb{R}^d$ be a compact set and let $C(K)$ denote the space of continuous functions on K endowed with the uniform norm

$$\|f\|_\infty = \sup_{x \in K} |f(x)|. \tag{E.8}$$

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous activation function that is not a polynomial. Then the class of functions of the form

$$f_N(x) = \sum_{i=1}^{N} a_i \, \sigma(w_i^\top x + b_i), \tag{E.9}$$

with parameters $a_i \in \mathbb{R}$, $w_i \in \mathbb{R}^d$, and $b_i \in \mathbb{R}$, is dense in $C(K)$. That is, for any $f \in C(K)$ and any $\varepsilon > 0$, there exists N and parameters $\{a_i, w_i, b_i\}_{i=1}^N$ such that

$$\sup_{x \in K} |f(x) - f_N(x)| < \varepsilon. \tag{E.10}$$

This result implies that feedforward neural networks with a single hidden layer possess the capacity to approximate any continuous function on compact domains arbitrarily well, provided a sufficient number of hidden units is available. Extensions of the theorem show that similar approximation properties hold in other function spaces, including L^p spaces and certain Sobolev spaces.

It is important to emphasize that universal approximation is an *existence* result. The theorem guarantees the existence of network parameters achieving a desired approximation accuracy, but it does not provide a constructive method for determining those parameters. In particular, the theorem does not address issues related to optimization, training efficiency, or generalization from finite datasets.

Despite these limitations, universal approximation results provide theoretical justification for the use of neural networks as flexible surrogate models in scientific computing and machine learning. In the context of scientific machine learning and physics-informed neural networks, these results support the use of neural networks as function approximators for unknown solution fields, nonlinear operators, or reduced dynamical representations arising in complex systems.

E.1.3 Training Objectives and Optimization

Training a neural network consists of identifying parameters that minimize a loss function measuring the discrepancy between predicted outputs and observed data. Let

$$\mathcal{D} = \{(x_k, y_k)\}_{k=1}^M \tag{E.11}$$

denote a dataset of input–output pairs, where $x_k \in \mathbb{R}^{d_0}$ and $y_k \in \mathbb{R}^{d_L}$. If $\hat{y}(x; \theta)$ denotes the network output with parameters θ, training can be formulated as the optimization problem

$$\min_{\theta \in \mathbb{R}^P} \mathcal{L}(\theta), \tag{E.12}$$

where $\mathcal{L}$ is a loss function defined over the dataset.

For regression problems, a common choice is the mean squared error (MSE)

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{M} \sum_{k=1}^{M} \|\hat{y}(x_k; \theta) - y_k\|_2^2. \tag{E.13}$$

To mitigate overfitting and improve conditioning, it is common to augment the data loss with a regularization term:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \lambda \Omega(\theta), \tag{E.14}$$

where $\lambda > 0$ is a regularization parameter. Typical choices include

$$\Omega(\theta) = \|\theta\|_2^2 \quad \text{(weight decay)} \tag{E.15}$$

or

$$\Omega(\theta) = \|\theta\|_1, \tag{E.16}$$

which promotes sparsity in the parameter vector.

The resulting optimization problem is typically high-dimensional and non-convex. In practice, training relies on iterative gradient-based algorithms [172, 222]. A basic gradient descent update takes the form

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t), \tag{E.17}$$

where $\eta > 0$ denotes the learning rate. The gradient $\nabla_\theta \mathcal{L}$ is efficiently computed using the backpropagation algorithm, which applies the chain rule through the layered structure of the network.

In large-scale problems, gradients are often approximated using randomly selected subsets of the training data, leading to stochastic gradient descent (SGD) and mini-batch variants. Momentum methods introduce a velocity variable that accumulates past gradients:

$$v_{t+1} = \mu v_t + \nabla_\theta \mathcal{L}(\theta_t), \tag{E.18}$$

$$\theta_{t+1} = \theta_t - \eta v_{t+1}, \tag{E.19}$$

where $\mu \in [0, 1)$ controls the contribution of previous updates. Variants such as Nesterov acceleration evaluate gradients at a look-ahead point and often yield improved convergence behavior. Adaptive algorithms adjust the learning rate individually for each parameter. One widely used example is the Adam optimizer [172, 222], which maintains exponential moving averages of the gradient and its squared magnitude:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{E.20}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t), \tag{E.21}$$

where $g_t = \nabla_\theta \mathcal{L}(\theta_t)$. After bias correction,

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{E.22}$$

the parameter update is given by

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{E.23}$$

Second-order optimization methods incorporate curvature information through the Hessian matrix

$$H(\theta) = \nabla_\theta^2 \mathcal{L}(\theta). \tag{E.24}$$

Newton-type methods update parameters according to

$$\theta_{t+1} = \theta_t - H(\theta_t)^{-1} \nabla_\theta \mathcal{L}(\theta_t). \tag{E.25}$$

Although these methods can exhibit rapid local convergence, computing and inverting the Hessian is often impractical for large neural networks. Consequently, quasi-Newton approaches and approximations such as the Levenberg–Marquardt algorithm are typically used in smaller-scale settings [223].

E.1.4 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) constitute a class of scientific machine learning methods that incorporate known physical laws directly into the training of neural networks. Introduced by Raissi et al. [184] and further developed in subsequent works [52], PINNs combine data-driven approximation with constraints arising from governing equations such as ordinary or partial differential equations.

Consider a physical system described by a differential equation

$$\mathcal{N}[u](x,t) = 0, \tag{E.26}$$

defined on a spatial domain $\Omega \subset \mathbb{R}^d$ and time interval $t \in [0,T]$. The operator $\mathcal{N}$ may represent a nonlinear differential operator involving spatial and temporal derivatives. The problem is typically complemented by boundary and initial conditions of the form

$$\mathcal{B}[u](x,t) = 0, \quad (x,t) \in \partial\Omega \times [0,T], \tag{E.27}$$

$$u(x,0) = u_0(x). \tag{E.28}$$

The objective is to approximate the unknown solution $u(x,t)$. In the PINN framework, the solution is represented by a neural network

$$u_\theta(x,t), \tag{E.29}$$

parameterized by weights and biases collected in θ. The network therefore acts as a parametric function approximator mapping input coordinates (x,t) to the predicted state. Instead of relying solely on labeled data, PINNs enforce that the neural network satisfies the governing equation. This is achieved by defining the residual

$$r_\mathcal{N}(x,t;\theta) = \mathcal{N}[u_\theta](x,t). \tag{E.30}$$

Spatial and temporal derivatives appearing in $\mathcal{N}$ are computed using automatic differentiation applied to the neural network representation. Similarly, residuals associated with boundary and initial conditions are defined as

$$r_\mathcal{B}(x,t;\theta) = \mathcal{B}[u_\theta](x,t), \tag{E.31}$$

$$r_\mathcal{I}(x;\theta) = u_\theta(x,0) - u_0(x). \tag{E.32}$$

The network parameters are obtained by minimizing a loss function that penalizes violations of the governing equations and constraints. A typical PINN objective takes the form

$$\mathcal{L}_{\text{PINN}}(\theta) = \frac{1}{N_r}\sum_{j=1}^{N_r} \|r_{\mathcal{N}}(x_j, t_j; \theta)\|_2^2 + \frac{1}{N_b}\sum_{j=1}^{N_b} \|r_{\mathcal{B}}(x_j, t_j; \theta)\|_2^2 + \frac{1}{N_i}\sum_{j=1}^{N_i} \|r_{\mathcal{I}}(x_j; \theta)\|_2^2, \tag{E.33}$$

where $\{(x_j, t_j)\}$ are collocation points sampled within the domain, on boundaries, or at initial times. Minimization of this loss enforces approximate satisfaction of the governing equations throughout the computational domain.

It is worth noting that PINNs can be used to solve both forward [184, 197, 224, 225, 226, 227, 228, 229] and inverse problems [45, 47, 184, 230, 231, 232, 233, 234, 235, 236], as well as for learning nonlinear functional operators [237, 238, 186, 239].

Despite their expressive power, neural networks present significant practical and theoretical challenges. Training deep architectures typically leads to highly nonconvex optimization problems characterized by numerous saddle points, flat regions, and ill-conditioned landscapes. Although overparameterization may facilitate optimization in certain regimes, convergence can still be slow or unstable and may depend strongly on initialization strategies and hyperparameter choices. Furthermore, high-capacity networks are capable of fitting training data extremely well, including noise, which makes effective generalization dependent on appropriate regularization, architectural inductive biases, data augmentation strategies, and careful validation procedures. The relationship between model complexity, data availability, and generalization performance therefore remains subtle, particularly in modern overparameterized learning regimes.

In addition to statistical considerations, neural network models often impose substantial computational demands. Large architectures require significant memory and computational resources, frequently necessitating specialized hardware such as GPUs or TPUs and careful engineering for efficient parallel execution. These requirements can be even more pronounced in scientific machine learning settings, where the evaluation of loss functions may involve complex differential operators, multiple interacting physical fields, or high-dimensional input spaces.

Optimization challenges are further exacerbated by issues such as exploding or vanishing gradients, especially in deep or recurrent architectures. Even in feedforward networks, different loss components may exhibit widely varying magnitudes, resulting in poorly conditioned optimization problems. In the context of PINNs, an additional difficulty arises from the need to balance multiple loss terms associated with governing equations, boundary conditions, and initial conditions. Improper weighting of these contributions may lead to slow convergence or inaccurate enforcement of the underlying physical constraints.

Finally, when applied to partial differential equations exhibiting sharp gradients, mul-

tiscale phenomena, or stiff dynamics, PINNs may struggle to accurately represent fine-scale structures without large network architectures and carefully designed sampling strategies. Moreover, the common practice of enforcing boundary and initial conditions through penalty terms can introduce trade-offs between accurately fitting data and satisfying physical laws. These limitations have motivated ongoing research on improved network architectures, adaptive sampling strategies, domain decomposition methods, and hybrid approaches that combine neural network models with classical numerical solvers.

Appendix F

Benchmark solutions of nonlinear functional equations

Abstract Here, we provide additional details for the benchmark numerical solutions of nonlinear functional equations for chapter 6 which sought specifically in training sets.

Figure (F.1) depicts the numerical approximation accuracy, measured as the difference between computed and analytical solutions for the training set, which is built using a grid of 15×15 nodes uniformly distributed. The results using the 6th-order power series expansion of both the nonlinear transformation and the right-hand side of the discrete model are displayed in subplots 3(a) and 3(b), respectively. As expected, the power series expansion yields zero error for the $T_2(x_1, x_2)$ component and demonstrates good approximation accuracy for the $T_1(x_1, x_2)$ component, but primarily in the vicinity of the linearization-relevant point at $[0, 0]$. Beyond this point, the numerical approximation deteriorates significantly, reaching the order of 10^0 at the grid edge, particularly close to the singular point. This situation could potentially undermine the precision with which the observer estimates the unmeasured state.

Figures (F.1)(c),(d) depict the approximation accuracy of the PINN, when trained in the entire domain (without the greedy approach). As observed, the approximation accuracy of the PINN remains rather poor, of the order of 1 in the vicinity of the singularity. Figures (G.1)(e),(f) illustrate the approximation error using PINN with the greedy-wise training approach. As shown, the performance of the PINN is significantly better compared to the one involving training across the entire domain. Furthermore, Figure (6.2) showcases the performance of the schemes for the test set, which consist of a grid of 20×20 Chebyshev-Lobatto distributed nodes. Please notice that these results are similar to those observed in the training set.

Regarding the second benchmark problem presented in Chapter 6. Figure (F.2) depicts the numerical approximation accuracy, as the difference between computed and analytical solutions (6.54), obtained by various schemes for the training set. This particular benchmark problem is more challenging as the gradients it contains (see Fig. 6.4) are notably bigger compared to the previous one.

Figures (F.2)(a),(b) demonstrate the results achieved with a 6th-order truncation in the power series expansion of the nonlinear transformation and the right-hand side of the discrete model. In this particular problem, the steep gradient emanating from the singular point at $x_1 = -1$ poses a significant challenge as mentioned before. The numerical approximation provided by the series expansion method is notably poor across the domain, particularly of the order of 10^1 at the grid's edge, as mentioned previously, near the singular point.

Furthermore, Figures (F.2)(c),(d) depict the outcomes obtained with the PINN approach to learn the transformation operator (6.54) across the entire domain without the greedy approach. The approximation accuracy of this scheme is relatively modest, especially in the vicinity of the singularity, approximately of the order of 10^0 for $T_1(x_1, x_2)$ and of the order of 10^1 for $T_2(x_1, x_2)$. Nevertheless, it outperforms the conventional Taylor series expansion method in terms of numerical error accuracy across the entire domain in general.

Figures (F.2)(e),(f) present the results obtained from the PINN implementation with the

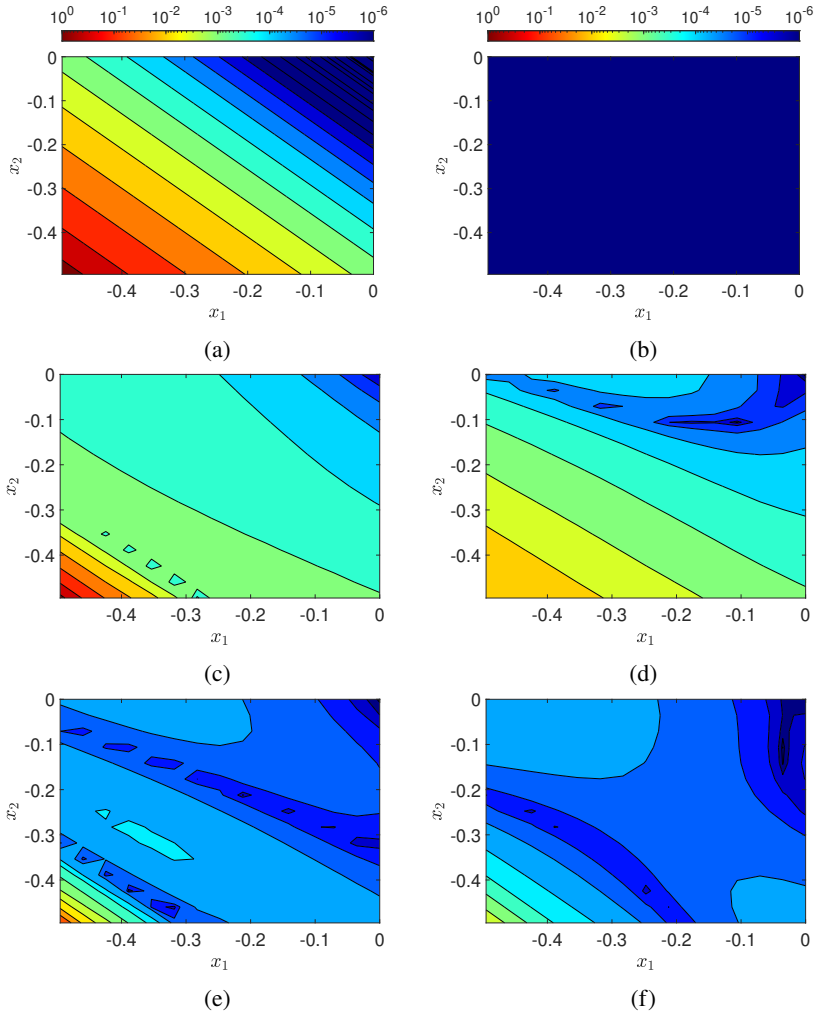


Figure F.1: Benchmark problem 1. Training sets (grids of 15×15 equispaced distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (6.36) in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PINN trained in the entire domain $[-0.495, 0] \times [-0.495, 0]$. (e),(f) PINN trained via the greedy-wise approach.

greedy approach. Notably, the adoption of the greedy-wise training method substantially enhances the numerical approximation accuracy when compared to PINN schemes trained across the entire domain in a single training step. Here, the numerical approxi-

mation error is of the order of 10^{-2} in the region close to the edge point $(-0.91, -0.91)$. Additionally, Figure (6.5) portrays the performance of the schemes when tested on a Chebyshev-Lobatto grid, with similar results observed for both the training and test sets.

Table (6.3) allows a comparison of the approximation errors between the analytical and numerical solutions. Specifically, the 6th order power series expansions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ with the two PINN solution approximations (with and without the greedy approach) are compared. To test the convergence of the schemes, we used a Chebyshev-Lobatto grid with 20×20 points. The reported errors include the median, 5th percentile, and 95th percentile aggregated over 100 runs.

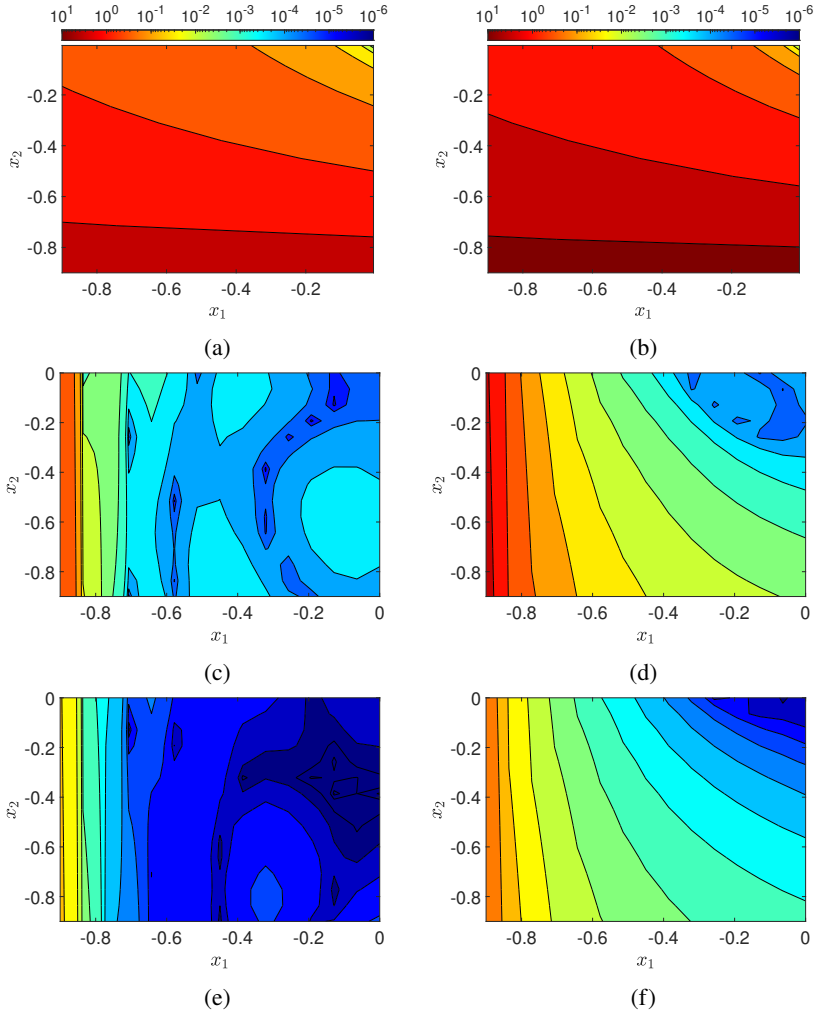


Figure F.2: Training sets (grids of 15×15 equispaced distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (6.3) in $[-0.91, 0] \times [-0.91, 0]$. (c),(d) PINN trained over the entire domain $[-0.91, 0] \times [-0.91, 0]$. (e),(f) PINN trained via the greedy-wise approach.

Appendix G

Supplementary Material for Chapter 7

Abstract This appendix reviews established numerical methods for solving systems of nonlinear functional equations, such as those introduced in Chapter 7. Two complementary approaches are discussed. First, we present a classical methodology based on Taylor series expansions, which relies on local analytical approximations of the solution. Second, we examine an alternative framework that exploits black-box simulators and reformulates the problem as an optimization task. In this setting, the solution is obtained as the minimizer of a suitably defined objective function, allowing the treatment of complex models for which explicit analytical expressions are unavailable. The material presented here serves as supplementary background to the methods discussed in Chapter 7.

G.1 Multivariate power-series expansion approach

In order to perform a comparative assessment of the performance of the proposed computational approach, a practical solution scheme for the associated system of NFE's (7.9) is needed. Since $f(x,u)$ as well as the solution $T(x)$ are all locally analytic around the origin, it is possible to calculate the solution in the form of a multivariate power-series. The proposed solution method involves the expansion of $f(x,u)$ as well as the unknown solution $T(x)$ in a power-series followed by equating the power coefficients of the same order of both sides of the NFE's (3). Such a procedure leads to a hierarchy of recursion formulas, through which one can calculate the N-th order power coefficients of $T(x)$, given the power coefficients of $T(x)$ up to the order $N-1$ (evaluated in previous recursive steps) [189].

In the derivation of the associated recursion formulas, it is quite convenient to employ

the following tensorial notation:

a) The entries of a constant matrix A are represented as a_i^j, where the subscript i refers to the corresponding row and the superscript j to the corresponding column of the matrix.

b) The partial derivatives of the μ-th component $f_\mu(x,u)$ of the vector function $f(x,u)$ with respect to the state variables x evaluated at $(x,u)=(0,0)$ are denoted as follows:

$$\begin{aligned} \hat{f}_\mu^i &= \frac{\partial f_\mu}{\partial x_i}(0,0) \\ \hat{f}_\mu^{ij} &= \frac{\partial^2 f_\mu}{\partial x_i \partial x_j}(0,0) \\ \hat{f}_\mu^{ijk} &= \frac{\partial^3 f_\mu}{\partial x_i \partial x_j \partial x_k}(0,0) \end{aligned} \tag{G.1}$$

etc., where $i, j, k, ..=1, ..., n$.

c) The partial derivatives of the μ-th component $f_\mu(x,u)$ of the vector function $f(x,u)$ with respect to the input variable u evaluated at $(x,u)=(0,0)$ are denoted as follows:

$$g_\mu^i = \frac{\partial^i f_\mu}{\partial u^i}(0,0) \tag{G.2}$$

etc.

d) The standard summation convention where repeated upper and lower tensorial indices are summed up.

Under the above notation the l-th component $T_l(x)$ of the unknown solution $T(x)$ can be expanded in a multivariate power series as follows:

$$\begin{aligned} T_l(x) &= \frac{1}{1!}T_l^{i_1}x_{i_1} + \frac{1}{2!}T_l^{i_1 i_2}x_{i_1}x_{i_2} + ... + \\ &+ \frac{1}{N!}T_l^{i_1 i_2 ... i_N}x_{i_1}x_{i_2}...x_{i_N} + ... \end{aligned} \tag{G.3}$$

As mentioned earlier, the proposed procedure is initiated by considering the expansion of the components of the vector function $f(x,u)$ in multivariate power-series. Substituting the power-series expansions of $T(x)$, $f(x,u)$ into (7.9) and matching the power coefficients of the same order, the following recursive relations are obtained:

First order terms; N=1

$$T_l^\mu(\hat{f}_\mu^{i_1} - g_\mu^i c^k T_k^{i_1}) = T_l^\mu \hat{f}_\mu^{i_1} - T_l^\mu g_\mu^i c^k T_k^{i_1} = a_l^\mu T_\mu^{i_1} \tag{G.4}$$

with: $i_1 = 1, ...n$ and $l = 1, ..n$. Note that under the matrix notation and the summation convention introduced above, the set of algebraic equations (A.4) can be recast into the

following matrix equation:

$$\bar{T}J - A\bar{T} = \bar{T}Gc\bar{T} \tag{G.5}$$

where the unknown matrix $\bar{T}$ in (A.5) is the Jacobian of the map $T(x)$ evaluated at the origin. Under the assumptions of Theorem 2.1, the unique invertible solution of the above quadratic matrix equation is given by : $\bar{T} = W^{-1}$, where W is the unique and invertible solution of the Lyapunov matrix equation shown below [189]:

$$JW - WA = Gc \tag{G.6}$$

Since Lyapunov equations of the above type can be solved using a software package such as Matlab/Maple, the calculation of the solution of the first-order algebraic equations (A.4) does not pose any challenges.

N-th order terms; $N \geq 2$

$$\sum_{L=1}^{N} \sum_{\substack{0\leq m_1\leq m_2\leq\cdots\leq m_L \\ m_1+m_2+\cdots+m_L=N}} T_l^{j_1\ldots j_L}\left(\hat{f}_{j_1}^{m_1}\ldots\hat{f}_{j_L}^{m_L} - \pi_{j_1}^{m_1}\ldots\pi_{j_L}^{m_L}\right) = a_l^{\mu}T_{\mu}^{i_1\ldots i_N} \tag{G.7}$$

where:

$$\pi_{jl}^{m_L} = \sum_{P=1}^{L} \sum_{\substack{0\leq n_1\leq n_2\leq\cdots\leq n_P \\ n_1+n_2+\cdots+n_P=m_L}} g_{jl}^{n_1}\, c^k\, T_k^{n_2\ldots n_P}. \tag{G.8}$$

with $i_1,\ldots,i_N = 1,\ldots,n$ and $l = 1,\ldots,n$. Notice that the second summation symbol in (A.7) indicates summing up the relevant quantities over the $\dfrac{N!}{m_1!\ldots m_L!}$ possible combinations to assign the N indices $(i_1,\ldots,i_N)$ as upper indices to the L positions: $\{\hat{f}_{j_1},\ldots\hat{f}_{j_L}\}$ and $\{\pi_{j_1},\ldots\pi_{j_L}\}$, with m_1 of them being put in the first position, m_2 of them in the second position , etc. ($\sum_{i=1}^{L} m_i = N$). Similar rules apply to equation (A.8). Please notice that equations (A.7,A.8) represent a set of linear algebraic equations in the unknown coefficients $T_{\mu}^{i_1,\ldots,i_N}$ for $N \geq 2$. Furthermore, it should be pointed out, that the above series solution method for the system of NFEs (7.9) may be accomplished in an automatic fashion by exploiting the computational capabilities of a symbolic software package such as Wolfram Mathematica and MAPLE.

G.2 Learning the Feedback Linearization Operator from Black-Box simulators

One of the main differences between this method and the previous one, is that in this method we are not expanding both sides of the NFEs (7.37), but just $T(x)$ in order to calculate the unknown coefficients of its series expansion through let's say nonlinear least squares. The information regarding the system is derived from or given by the output

of the black-box simulator. Therefore, the problem under consideration can be stated as one whose target is finding the values of the vector h such that the sum of squared errors on the discretization mesh is minimized in some norm for instance the 2-norm, i.e.

$$\min_{h} \sum_{i=1}^{N} \parallel R_i(h) \parallel_2^2 \tag{G.9}$$

where the vector function $R_i(h)$ is defined as follows:

$$R_i(h) = \hat{T}(f(x_i, -c\hat{T}(x_i); h) - A\hat{T}(x_i; h), \quad \forall x_i \tag{G.10}$$

As mentioned earlier, this nonlinear optimization problem can be solved using a Gauss-Newton method or the Levenberg Marquard method in an iterative fashion, by enforcing equality (G.10) at every point of the discretized mesh. For example, for the power-series expansion the algorithm for computing the transformation law using a black-box simulator reads as follows:

- Choose a subdomain of the solution state space of the nonlinear system, i.e. $D \subseteq \mathbb{R}^n$ in a mesh of $N \times N$ points. In such a domain, the solution of the NFEs system (on the basis of which the feedback controller itself is synthesized) will be learned as well.

- Expand the transformation map $T(x)$ in a power-series up to order p around the equilibrium x_o, meaning that $T(x)$ must be expressed as a function of the vector x and also the power-series coefficients $h \in \mathbb{R}^m$, i.e., $\hat{T}(x, h)$, i.e.

$$\begin{aligned}\hat{T}_1(x_{i=1,\cdots,n}; h_{j,k=1,\cdots,p}) = {} & h_{1,1}x_1 + h_{1,2}x_2 + \frac{1}{2!}h_{1,3}x_1^2 + \frac{1}{2!}h_{1,4}x_2^2 \\ & + h_{1,5}x_1x_2 + \cdots + O_1(p+1)\end{aligned} \tag{G.11}$$

$$\begin{aligned}\hat{T}_2(x_{i=1,\cdots,n}; h_{j,k=1,\cdots,p}) = {} & h_{2,1}x_1 + h_{2,2}x_2 + \frac{1}{2!}h_{2,3}x_1^2 + \frac{1}{2!}h_{2,4}x_2^2 \\ & + h_{2,5}x_1x_2 + \cdots + O_2(p+1)\end{aligned} \tag{G.12}$$

$$\vdots$$

$$\begin{aligned}\hat{T}_n(x_{i=1,\cdots,n}; h_{j,k=1,\cdots,p}) = {} & h_{n,1}x_1 + h_{n,2}x_2 + \frac{1}{2!}h_{n,3}x_1^2 + \frac{1}{2!}h_{n,4}x_2^2 \\ & + h_{n,5}x_1x_2 + \cdots + O_n(p+1)\end{aligned} \tag{G.13}$$

  Then, write the feedback control law as $u = -c\hat{T}(x_i)$.

- Obtain the information of the system by calling the output of the black-box simulator using the series expansion up to order p of $T(x)$ as u.

- Construct both sides of the NFEs system, and get a residual as indicated in (G.10).

- Add to the residual

$$R_i(h) = \hat{T}(f(x_i, -c\hat{T}(x_i); h) - A\hat{T}(x_i; h) = 0 \tag{G.14}$$

the following conditions:

  - Initial condition

$$\hat{T}_j(0) = 0, \quad j, k = 1, 2, \ldots, n \tag{G.15}$$

  - Derivative of $\hat{T}(x_i)$ evaluated the equilibrium $x_o = 0$

$$\frac{\partial \hat{T}_j}{\partial x_k}(0) - \frac{\partial T_j}{\partial x_k}(0) = 0, \quad j, k = 1, 2, \ldots, n \tag{G.16}$$

where, as mentioned before, $\frac{\partial T_j}{\partial x_k}(0)$ is the (j, k)-th element of the Jacobian matrix of $T(x)$ computed at the equilibrium and obtained by solving equation (6.26). The latter serves as a pinning condition for the optimization problem to find the best coefficients for the series expansion of the transformation map $T(x)$ that satisfy the residual $R_i(h) = 0$ equality. Indeed, without this pinning condition, it is also probable that the optimization process finds the trivial solution that also satisfies these properties, but of course does not represent the feedback controller since the trivial solution maps all the states to the kernel of the linearized space. Finally, the derivative of $T(x)$ can be computed, for instance using finite differences.

- Compute the unknown coefficients of $T(x_i; h)$ using a nonlinear optimization algorithm, such as the Levenberg–Marquardt, Gauss-Newton or perhaps using an unconstrained optimization algorithm, such as the Broyden, Fletcher, Goldfarb, Shanno (BFGS) method.

A similar procedure can be used for the Physics Informed Machine-Learning (PIML) scheme.

G.3 Analytical Derivatives for Training

For completeness, we provide the analytical expressions of the derivatives required to evaluate the Jacobian $\frac{\partial \hat{T}}{\partial x}$ and the parameter gradients entering (7.31). While in modern software environments these quantities can be obtained through automatic differentiation, explicit formulas are useful both for implementation in environments without automatic differentiation and for the black-box simulator setting where derivatives may be approximated numerically.

Derivatives with Respect to the Input

Let $x = (x_1, \ldots, x_d)$. The derivative with respect to the k-th component of the first hidden layer activation is

$$\frac{\partial}{\partial x_k}\phi_s^{(1)}\left(\sum_{h=1}^{d} W_{hs}^{(1)} x_h + \beta_s^{(1)}\right) = W_{ks}^{(1)}\phi_s^{(1)\prime}\left(\sum_{h=1}^{d} W_{hs}^{(1)} x_h + \beta_s^{(1)}\right). \tag{G.17}$$

Equation (G.17) can be written in matrix form as

$$\frac{\partial \Phi_1(W^{(1)T}x + \beta^{(1)})}{\partial x_k} = {W_k}^{(1)T} \odot \Phi_1'(W^{(1)T}x + \beta^{(1)}), \tag{G.18}$$

where Φ_1' denotes the vector of derivatives of the corresponding activation functions, $W_k^{(1)}$ is the k-th row of $W^{(1)}$, and $\odot$ denotes the Hadamard (element-wise) product.

For the second hidden layer, the derivative of the composed activation is

$$\frac{\partial}{\partial x_k}\phi_i^{(2)}\Big(\eta_i(x)\Big) = \phi_i^{(2)\prime}\Big(\eta_i(x)\Big) \times \sum_{s=1}^{N_1} W_{si}^{(2)} W_{ks}^{(1)} \phi_s^{(1)\prime}\left(\sum_{h=1}^{d} W_{hs}^{(1)} x_h + \beta_s^{(1)}\right), \tag{G.19}$$

$$\eta_i(x) = \sum_{s=1}^{N_1} W_{si}^{(2)} \phi_s^{(1)}\left(\sum_{h=1}^{d} W_{hs}^{(1)} x_h + \beta_s^{(1)}\right) + \beta_i^{(2)}. \tag{G.20}$$

In matrix form,

$$a^{(1)}(x) = W^{(1)T}x + \beta^{(1)}, \qquad a^{(2)}(x) = W^{(2)T}\Phi_1(a^{(1)}(x)) + \beta^{(2)}. \tag{G.21}$$

$$\frac{\partial \Phi_2(a^{(2)}(x))}{\partial x_k} = \Phi_2'(a^{(2)}(x)) \odot \Big(W^{(2)T}\big(W_k^{(1)T} \odot \Phi_1'(a^{(1)}(x))\big)\Big). \tag{G.22}$$

Finally, the (j,k) element of the Jacobian matrix $\frac{\partial \hat{T}}{\partial x}$ is

$$\begin{aligned}\frac{\partial}{\partial x_k}\hat{T}_j(x) = \sum_{i=1}^{N_2} W_{ij}^{(o)} \phi_i^{(2)\prime}\left(\sum_{s=1}^{N_1} W_{si}^{(2)} \phi_s^{(1)}\left(\sum_{h=1}^{d} W_{hs}^{(1)} x_h + \beta_s^{(1)}\right) + \beta_i^{(2)}\right) \\ \times \left(\sum_{s=1}^{N_1} W_{si}^{(2)} W_{ks}^{(1)} \phi_s^{(1)\prime}\left(\sum_{h=1}^{d} W_{hs}^{(1)} x_h + \beta_s^{(1)}\right)\right).\end{aligned} \tag{G.23}$$

Equivalently, in matrix form,

$$\frac{\partial \hat{T}_j}{\partial x_k} = W^{j(o)T}\Big(\Phi_2'(a^{(2)}(x)) \odot \big(W^{(2)T}(W_k^{(1)T} \odot \Phi_1'(a^{(1)}(x)))\big)\Big), \tag{G.24}$$

where $W^{j(o)}$ is the j-th column of $W^{(o)}$.

Derivatives with Respect to Network Parameters

The derivatives of the residuals with respect to a generic parameter $p \in P$ entering (7.31) are written as

$$\begin{aligned}
\frac{\partial r_{ij}^{(1)}}{\partial p} &= \frac{\partial \hat{T}_j\big(f(x_i, -c^T\hat{T}(x_i))\big)}{\partial p}\frac{\partial f(x_i, -c^T\hat{T}(x_i))}{\partial u}\cdot\Big(-c^T\frac{\partial \hat{T}(x_i)}{\partial p}\Big) - {\alpha_j}^T\frac{\partial \hat{T}(x_i)}{\partial p},\\
\frac{\partial r_j^{(2)}}{\partial p} &= \frac{\partial \hat{T}_j(0)}{\partial p},\\
\frac{\partial r_{jk}^{(3)}}{\partial p} &= \frac{\partial^2 \hat{T}_j}{\partial p \partial x_k}(0), \qquad i = 1,\dots,M, \quad j,k = 1,\dots,d, \quad p \in P.
\end{aligned} \tag{G.25}$$

We now list the explicit partial derivatives of $\hat{T}_j(x)$ with respect to the weights and biases.

- For $p = W_{hq}^{(o)}$:

$$\frac{\partial \hat{T}_j(x)}{\partial W_{hq}^{(o)}} = \begin{cases} \phi_h^{(2)}\Big(\sum_{s=1}^{N_1} W_{sh}^{(2)}\phi_s^{(1)}\Big(\sum_{k=1}^{d} W_{ks}^{(1)}x_k + \beta_s^{(1)}\Big) + \beta_h^{(2)}\Big), & \text{if } q = j,\\ 0, & \text{if } q \neq j. \end{cases} \tag{G.26}$$

- For $p = W_{hq}^{(2)}$:

$$\frac{\partial \hat{T}_j(x)}{\partial W_{hq}^{(2)}} = W_{hj}^{(o)}\phi_h^{(2)\prime}\Big(\sum_{s=1}^{N_1} W_{sh}^{(2)}\phi_s^{(1)}\Big(\sum_{k=1}^{d} W_{ks}^{(1)}x_k + \beta_s^{(1)}\Big) + \beta_h^{(2)}\Big)\phi_q^{(1)}\Big(\sum_{k=1}^{d} W_{ks}^{(1)}x_k + \beta_s^{(1)}\Big). \tag{G.27}$$

- For $p = W_{hq}^{(1)}$:

$$\begin{aligned}
\frac{\partial \hat{T}_j(x)}{\partial W_{hq}^{(1)}} = &\sum_{i=1}^{N_2} W_{ij}^{(o)}\phi_i^{(2)\prime}\Big(\sum_{s=1}^{N_1} W_{si}^{(2)}\phi_s^{(1)}\Big(\sum_{k=1}^{n} W_{ks}^{(1)}x_k + \beta_s^{(1)}\Big) + \beta_i^{(2)}\Big)\\
&\times\Big(W_{hi}^{(2)}x_q\phi_h^{(1)\prime}\Big(\sum_{k=1}^{n} W_{kh}^{(1)}x_k + \beta_h^{(1)}\Big)\Big).
\end{aligned} \tag{G.28}$$

- For $p = \beta_h^{(o)}$:

$$\frac{\partial \hat{T}_j(x)}{\partial \beta_h^{(o)}} = \begin{cases} 1, & \text{if } h = j,\\ 0, & \text{if } h \neq j. \end{cases} \tag{G.29}$$

- For $p = \beta_h^{(2)}$:

$$\frac{\partial \hat{T}_j(x)}{\partial \beta_h^{(2)}} = W_{hj}^{(o)} \phi_h^{(2)} \left(\sum_{s=1}^{N_1} W_{sh}^{(2)} \phi_s^{(1)} \left(\sum_{k=1}^{d} W_{ks}^{(1)} x_k + \beta_s^{(1)} \right) + \beta_h^{(2)} \right). \tag{G.30}$$

- For $p = \beta_h^{(1)}$:

$$\begin{aligned} \frac{\partial \hat{T}_j(x)}{\partial \beta_h^{(1)}} = & \sum_{i=1}^{N_2} W_{ij}^{(o)} \phi_i^{(2)'} \left(\sum_{s=1}^{N_1} W_{si}^{(2)} \phi_s^{(1)} \left(\sum_{k=1}^{d} W_{ks}^{(1)} x_k + \beta_s^{(1)} \right) + \beta_i^{(2)} \right) \\ & \times \left(W_{hi}^{(2)} \phi_h^{(1)} \left(\sum_{k=1}^{d} W_{kh}^{(1)} x_k + \beta_h^{(1)} \right) + \beta_i^{(2)} \right). \end{aligned} \tag{G.31}$$

As noted above, these expressions may be evaluated analytically, computed via automatic differentiation, or approximated numerically (e.g., centered finite differences) when only a black-box simulator is available.

G.4 Numerical results for model explicitly available

Figure (G.1) shows the numerical approximation accuracy (difference between computed and analytical solutions) obtained by the various schemes for the training set. Figures (G.1)(a),(b) show the results obtained with a $6th$ order power-series expansion of the nonlinear transformation and the right-hand-side of the discrete model. As expected, the power-series expansion results in zero error for the $T_2(x_1, x_2)$ component, and in a good approximation accuracy for the $T_1(x_1, x_2)$ component, but only in the region close to the linearization-relevant point, which in this case is $[0, 0]$, while away from it, the numerical approximation is poor (of the order of 10^0 at the edge of the grid) close to the singular point. Figures (G.1)(c),(d) depict the results obtained with the PIML implemented in Tensorflow using the BFGS optimizer for learning the non-linear transformation law in the entire domain. The approximation accuracy of the scheme is rather poor, especially near the singularity (of the order of 10^0). Figures (G.1)(e),(f) depict the results obtained from the PIML implemented with the Tensorflow and the BFGS optimizer using the greedy training procedure. Figures (G.1)(g),(h) depict the results obtained with the PIML implemented in Matlab using the LM optimizer and the greedy training procedure. It is evident that the greedy training procedure results in significantly enhanced numerical approximation accuracy compared to the PIML schemes trained in the entire domain at once. In particular, the PIML scheme implemented in TensorFlow resulted in a numerical approximation error of the order of 10^{-2} and the home-made Matlab resulted in a numerical approximation error of the order of 10^{-3}, in the region close to the edge point $(-0.495, -0.495)$. In addition, Table G.1 detail the numerical approximation accuracy of the various schemes fo the train sets, respectively. Taking denser grids and more neurons in each hidden layer, did not change qualitatively the numerical approximation

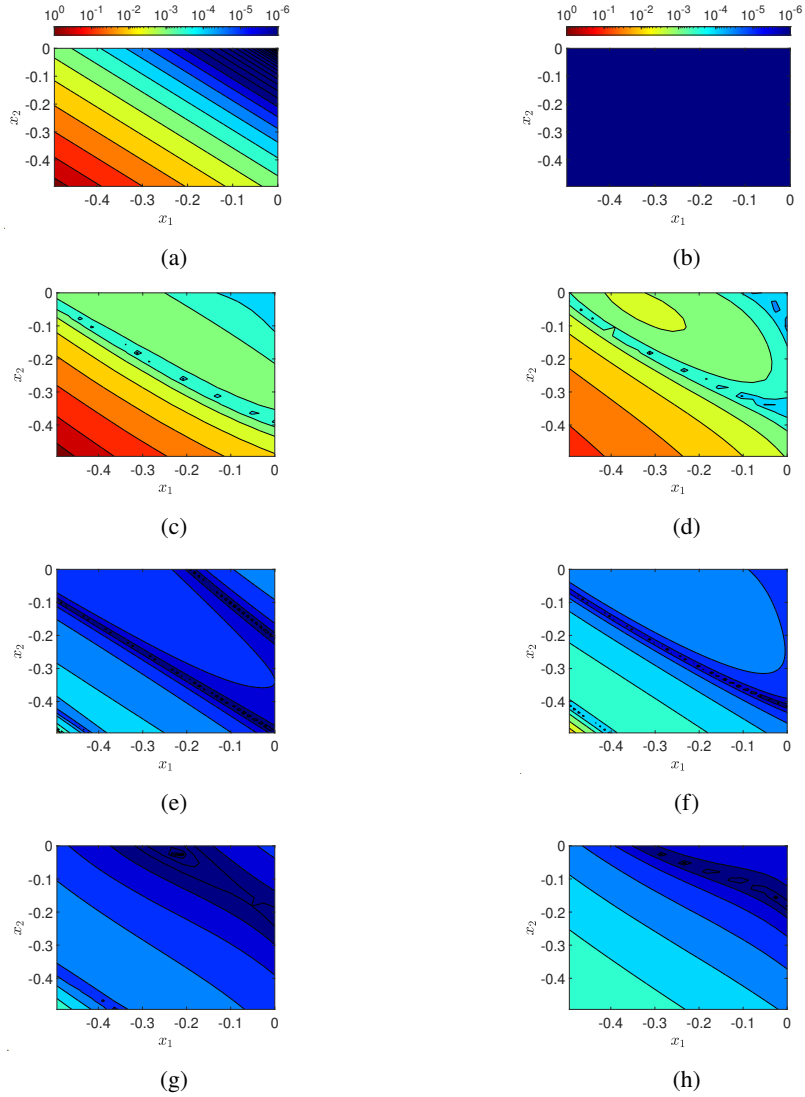


Figure G.1: Model explicitly available. Training sets (grids of 20×20 equispaced distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (7.35) in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PIML in Tensorflow trained in the entire domain. (e),(f) PIML in Tensorflow trained via the greedy approach. (g),(h) PIML in Matlab trained via the greedy approach.

Table G.1: Model explicitly available. Training sets (grids of 20×20 equispaced distributed points). Error norms (L_1, L_2 and L_∞) between the analytical and computed solution of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ using the various schemes trained both greedy and in the entire domain $[-0.495, 0] \times [-0.495, 0]$.

Transformation $T(x)$	Error norm	power-series $6th$ order	PIML(TF) Entire domain	PIML(Matlab) Greedy	PIML(TF) Greedy
$T_1(x_1, x_2)$	$\|\cdot\|_1$	6.76E+01	6.28E+00	2.03E−03	3.65E−02
	$\|\cdot\|_2$	3.62E+00	3.73E+00	1.12E−03	3.26E−02
	$\|\cdot\|_\infty$	1.21E+00	2.81E+00	1.05E−03	3.10E−03
$T_2(x_1, x_2)$	$\|\cdot\|_1$	0	1.40E+00	6.33E−03	6.61E−02
	$\|\cdot\|_2$	0	1.00E+00	1.40E−03	3.68E−02
	$\|\cdot\|_\infty$	0	5.94E−01	6.73E−04	1.00E−02

accuracy. Indicatively, in Fig (G.3), we depict the numerical approximation accuracy obtained with the PIML implemented in TensorFlow (TD) trained with BFGS in the entire domain $[-0.495, 0] \times [-0.495, 0]$ using 100×100 equispaced points and different number of neurons in each hidden layer. In particular, Figs. (G.3)(a),(b) show the numerical approximation accuracy in the training set using two hidden layers with five neurons in each layer, Fig. (G.3)(c),(d) with ten neurons, and Figs. (G.3)(e),(f) with fifteen neurons in each layer. Finally, Fig. (G.4), reports the corresponding numerical approximation accuracy plots for the test set (a grid of Chebyshev-distributed points in $[-0.495, 0] \times [-0.495, 0]$).

As Figures (G.2)(a),(b) show, in the case of the black-box simulator, the power-series expansion attained a lower/worse numerical approximation accuracy level compared to the one when the model is assumed to be explicitly known (Figure (G.1)(a),(b)). The approximation error is rather poor (of the order of 10^0) in the region close to the singularity, i.e, the point $[-0.495, -0.495]$. Figures (G.2)(c),(d) depict the PIML scheme as implemented on Matlab trained in the entire domain. Here the approximation error is of the order of 10^{-1} in the region close to the singular point. Figures (G.2)(e),(f) depict the PIML scheme implemented on Matlab trained with the greedy procedure. The numerical approximation error in the domain where the step gradient appears is of the order of 10^{-3}, thus outperforming the power-series expansion approximation of the transformation law, as well as the PIML trained in the entire domain at once. It should be noted that the numerical approximation error obtained in the region close to the singular point $[-0.495, -0.495]$ is of the same order as that obtained with the PIML implemented when the model is assumed to be explicitly known. Finally, Figure (7.4) depicts the numerical approximation errors in the test set. The results are similar both for the training and test sets. Table G.2 detail the numerical approximation accuracy of the various schemes, for the training.

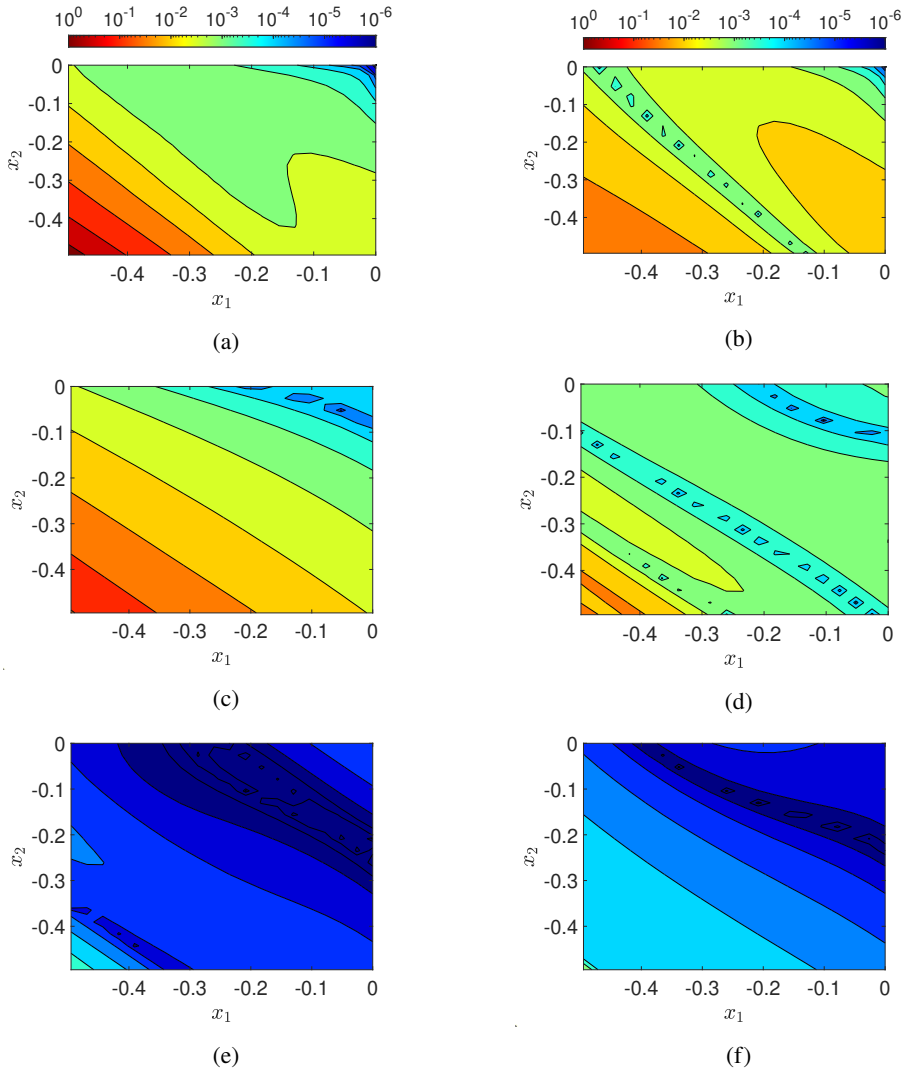


Figure G.2: Black-box simulator. Training sets (grids of 20×20 equispaced distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) $6th$ order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PIML in Matlab trained in the entire domain. (e),(f) PIML in Matlab trained via the greedy approach.

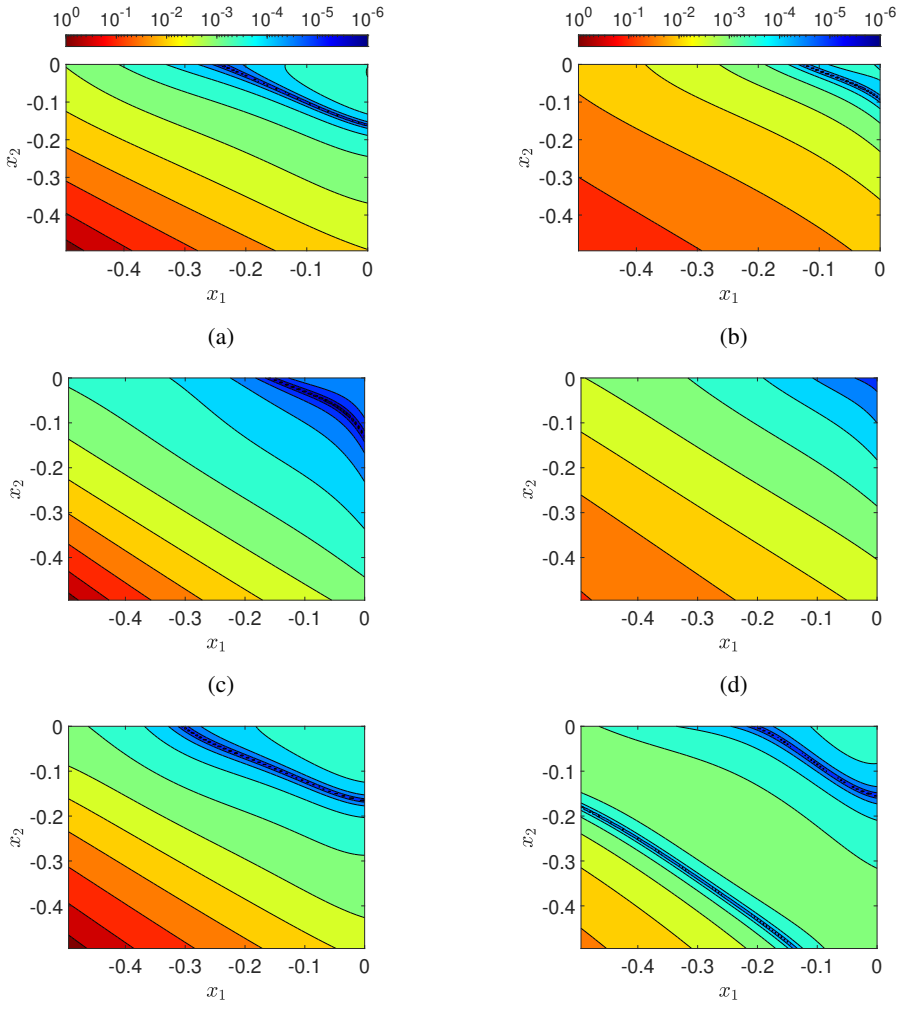


Figure G.3: Model explicitly available. Training set (grid of 100×100 equispaced distributed points in $[-0.495, 0] \times [-0.495, 0]$). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the Keras API in TensorFlow; training was performed in the entire domain. (a),(b) PIML, two hidden layers, five neurons in each layer. (c),(d) PIML, two hidden layers, ten neurons in each layer. (e),(f) PIML, two hidden layers, fifteen neurons in each layer.

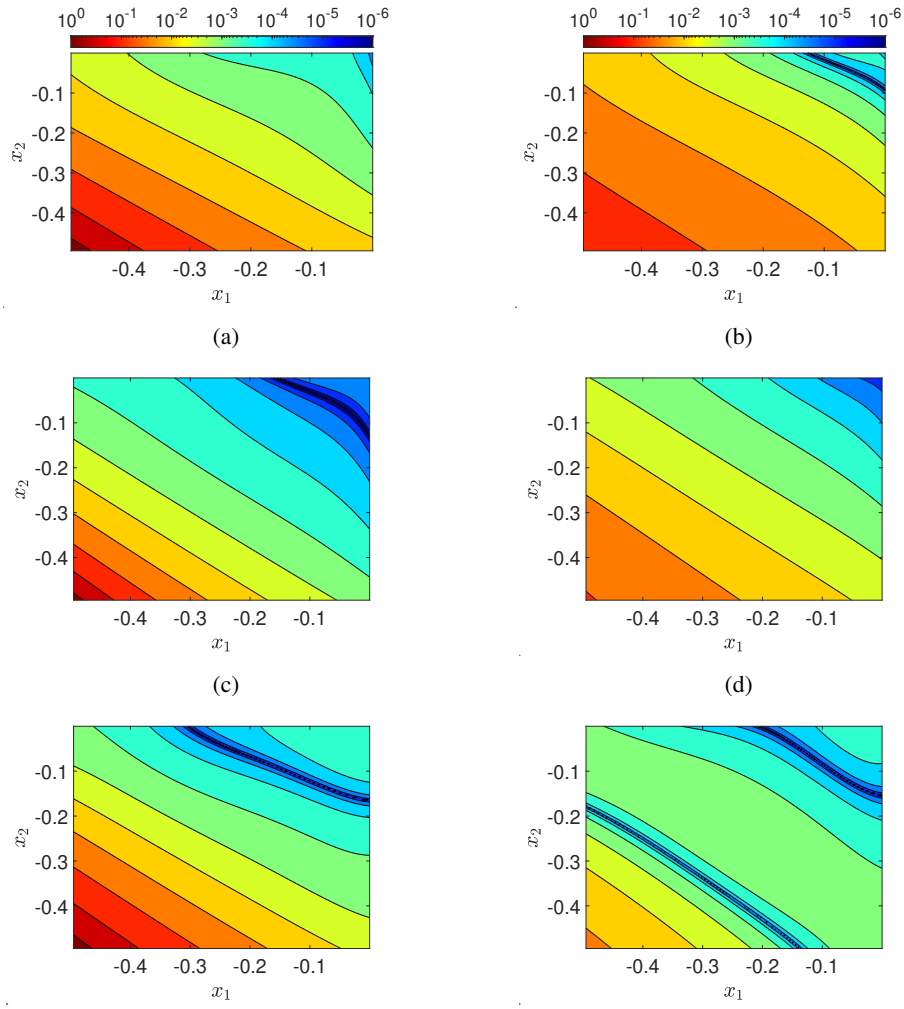


Figure G.4: Model explicitly available. Test set (grid of 150×150 Chebyshev-distributed points in $[-0.495, 0] \times [-0.495, 0]$). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the Keras API in TensorFlow; training was performed in the entire domain. (a),(b) PIML, two hidden layers, five neurons in each layer. (c),(d) PIML, two hidden layers, ten neurons in each layer. (e),(f) PIML, two hidden layers, fifteen neurons in each layer.

Table G.2: Black-box simulator. Training sets (grids of 20×20 equispaced distributed points). Error norms (L_1, L_2 and L_∞) between the analytical and computed solution of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ using the various schemes trained both greedy and in the entire domain $[-0.495, 0] \times [-0.495, 0]$.

Transformation $T(x)$	Error norm	power-series $6th$ order	PIML(Matlab) Entire domain	PIML(Matlab) Greedy
$T_1(x_1, x_2)$	$\|\cdot\|_1$	1.50E+01	1.21E+00	2.86E−03
	$\|\cdot\|_2$	9.73E+00	7.84E−01	2.78E−03
	$\|\cdot\|_\infty$	1.27E+00	1.35E−01	1.74E−03
$T_2(x_1, x_2)$	$\|\cdot\|_1$	4.95E−01	2.36E−01	7.03E−03
	$\|\cdot\|_2$	4.09E−01	2.44E−01	6.54E−03
	$\|\cdot\|_\infty$	3.07E−01	1.10E−01	4.82E−03

Bibliography

[1] J. Ladyman, J. Lambert, and K. Wiesner, "What is a complex system?" *European Journal for Philosophy of Science*, vol. 3, no. 1, pp. 33–67, 2013.

[2] M. E. J. Newman, "Complex systems: A survey," *arXiv preprint arXiv:1112.1440*, 2011.

[3] F. Bullo, *Lectures on Network Systems*. Kindle Direct Publishing, 2024, version 1.7. [Online]. Available: https://fbullo.github.io/lns

[4] M. Mitchell, *Complexity: A Guided Tour*. Oxford University Press, 2009.

[5] J. H. Holland, *Complexity: A Very Short Introduction*. Oxford University Press, 2014.

[6] G. Nicolis and I. Prigogine, *Self-Organization in Nonequilibrium Systems*. Wiley, 1977.

[7] H. Haken, *Synergetics: An Introduction*. Springer, 1983.

[8] ——, *Information and Self-Organization: A Macroscopic Approach to Complex Systems*. Springer, 2006.

[9] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2018.

[10] Y. Holovatch, R. Kenna, and S. Thurner, "Complex systems: physics beyond physics," *arXiv preprint arXiv:1610.01002*, 2016.

[11] N. Bellomo, D. Knopoff, and J. Soler, "Complex systems and collective behavior: From microscopic to macroscopic modeling," *Physics of Life Reviews*, vol. 42, pp. 1–26, 2022.

[12] A. J. Chorin, *Discrete Approach to Turbulence*. Springer, 2013.

[13] N. Bellomo and C. Dogbe, "Modeling crowd dynamics from a complex system viewpoint," *Mathematical Models and Methods in Applied Sciences*, vol. 22, no. supp01, p. 1230004, 2011.

[14] N. Bellomo, D. Burini, L. Gibelli, N. Outada, M. Twarogowska, and T. Wanka, "Active particles, modeling complex systems and social dynamics," *Mathematical Models and Methods in Applied Sciences*, vol. 27, no. 01, pp. 123–151, 2017.

[15] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, 1995.

[16] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, Sep 2000.

[17] A. Corbetta, J. A. Meeusen, C.-m. Lee, R. Benzi, and F. Toschi, "Physics-based modeling and data representation of pairwise interactions among pedestrians," *Physical Review E*, vol. 98, no. 6, p. 062310, 2018.

[18] X. Yang, H. Dong, Q. Wang, Y. Chen, and X. Hu, "Guided crowd dynamics via modified social force model," *Physica A: Statistical Mechanics and its Applications*, vol. 411, pp. 63–73, 2014.

[19] K. Nishidate, M. Baba, and R. J. Gaylord, "Cellular automaton model for random walkers," *Physical Review Letters*, vol. 77, no. 9, p. 1675, 1996.

[20] V. J. Blue and J. L. Adler, "Cellular automata microsimulation of bidirectional pedestrian flows," *Transportation Research Record*, vol. 1678, no. 1, pp. 135–141, 1999.

[21] F. Dietrich, G. Köster, M. Seitz, and I. von Sivers, "Bridging the gap: From cellular automata to differential equation models for pedestrian dynamics," *Journal of Computational Science*, vol. 5, no. 5, pp. 841–846, 2014.

[22] M. Kinateder, T. D. Wirth, and W. H. Warren, "Crowd dynamics in virtual reality," in *Crowd Dynamics, Volume 1: Theory, Models, and Safety Problems*, N. Bellomo and L. Gibelli, Eds. Springer, 2018, pp. 15–36, vision-based pedestrian model from Brown University.

[23] B. Aylaj, N. Bellomo, L. Gibelli, and A. Reali, "A unified multiscale vision of behavioral crowds," *Mathematical Models and Methods in Applied Sciences*, vol. 30, no. 01, pp. 1–22, 2020.

[24] N. Bellomo and C. Dogbe, "On the modelling crowd dynamics from scaling to hyperbolic macroscopic models," *Mathematical Models and Methods in Applied Sciences*, vol. 18, no. supp01, pp. 1317–1345, 2008.

[25] E. Cristiani, B. Piccoli, and A. Tosin, *Multiscale Modeling of Pedestrian Dynamics*. Springer, 2014.

[26] N. Bellomo and A. Bellouquid, "On multiscale models of pedestrian crowds from mesoscopic to macroscopic," *Commun. Math. Sci*, vol. 13, no. 7, pp. 1649–1664, 2015.

[27] N. Bellomo and C. Dogbé, "On the modeling of traffic and crowds: A survey of models, speculations, and perspectives," *SIAM Review*, vol. 53, no. 3, pp. 409–463, 2011.

[28] C. Dogbe, "On the modelling of crowd dynamics by generalized kinetic models," *Journal of Mathematical Analysis and Applications*, vol. 387, no. 2, pp. 512–532, 2012.

[29] D. Kim and A. Quaini, "A kinetic theory approach to model pedestrian dynamics in bounded domains with obstacles," *arXiv preprint arXiv:1901.07620*, 2019.

[30] S. Hoogendoorn and P. H. Bovy, "Gas-kinetic modeling and simulation of pedestrian flows," *Transportation Research Record*, vol. 1710, no. 1, pp. 28–36, 2000.

[31] B. Aylaj, N. Bellomo, L. Gibelli, and D. Knopoff, *Crowd dynamics by kinetic theory modeling: complexity, modeling, simulations, and safety*. Morgan & Claypool Publishers, 2020.

[32] N. Bellomo, J. Liao, A. Quaini, L. Russo, and C. Siettos, "Human behavioral crowds review, critical analysis and research perspectives," *Mathematical Models and Methods in Applied Sciences*, vol. 33, no. 08, pp. 1611–1659, 2023.

[33] M. J. Lighthill and G. B. Whitham, "On kinematic waves ii. a theory of traffic flow on long crowded roads," *Proceedings of the Royal Society of London A*, vol. 229, no. 1178, pp. 317–345, 1955.

[34] P. I. Richards, "Shock waves on the highway," *Operations Research*, vol. 4, no. 1, pp. 42–51, 1956.

[35] H. J. Payne, "Models of freeway traffic and control," *Simulation Councils Proceedings Series*, vol. 1, pp. 51–61, 1971.

[36] A. Aw and M. Rascle, "Resurrection of "second order" models of traffic flow," *SIAM Journal on Applied Mathematics*, vol. 60, no. 3, pp. 916–938, 2000.

[37] R. L. Hughes, "A continuum theory for the flow of pedestrians," *Transportation Research Part B: Methodological*, vol. 36, no. 6, pp. 507–535, 2002.

[38] R. M. Colombo and N. Pogodaev, "Nonlocal crowd dynamics models for several populations," *Journal of Nonlinear Science*, vol. 22, no. 4, pp. 511–547, 2012.

[39] E. Cristiani, F. S. Priuli, and A. Tosin, "Modeling rationality to control self-organization of crowds: an environmental approach," *SIAM Journal on Applied Mathematics*, vol. 75, no. 2, pp. 605–629, 2015.

[40] G. Albi, M. Bongini, E. Cristiani, and D. Kalise, "Invisible control of self-organizing agents leaving unknown environments," *SIAM Journal on Applied Mathematics*, vol. 76, no. 4, pp. 1683–1710, 2016.

[41] G. Albi, E. Cristiani, L. Pareschi, and D. Peri, "Mathematical models and methods for crowd dynamics control," *Crowd Dynamics, Volume 2: Theory, Models, and Applications*, pp. 159–197, 2020.

[42] F. Auletta, D. Fiore, M. J. Richardson, and M. di Bernardo, "Herding stochastic autonomous agents via local control rules and online target selection strategies," *Autonomous Robots*, vol. 46, no. 3, pp. 469–481, 2022.

[43] I. Panagiotopoulos, J. Starke, and W. Just, "Control of collective human behavior: Social dynamics beyond modeling," *Physical Review Research*, vol. 4, no. 4, p. 043190, 2022.

[44] X. Gong, M. Herty, B. Piccoli, and G. Visconti, "Crowd dynamics: Modeling and control of multiagent systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, no. 1, pp. 261–282, 2023.

[45] S. Lee, Y. M. Psarellis, C. I. Siettos, and I. G. Kevrekidis, "Learning black- and gray-box chemotactic pdes/closures from agent based monte carlo simulation data," *Journal of Mathematical Biology*, vol. 87, no. 1, p. 15, 2023.

[46] Y. M. Psarellis, S. Lee, T. Bhattacharjee *et al.*, "Data-driven discovery of chemotactic migration of bacteria via coordinate-invariant machine learning," *BMC Bioinformatics*, vol. 25, no. 1, p. 337, 2024.

[47] G. Fabiani, N. Evangelou, T. Cui, J. M. Bello-Rivas, C. P. Martin-Linares, C. Siettos, and I. G. Kevrekidis, "Task-oriented machine learning surrogates for tipping points of agent-based models," *Nature Communications*, vol. 15, no. 1, p. 4117, 2024.

[48] P. Degond and S. Motsch, "Continuum limit of self-driven particles with orientation interaction," *Mathematical Models and Methods in Applied Sciences*, vol. 18, no. suppl, pp. 1193–1215, 2008.

[49] J. A. Carrillo, M. Fornasier, G. Toscani, and F. Vecil, "Particle, kinetic, and hydrodynamic models of swarming," *Mathematical Models and Methods in Applied Sciences*, vol. 20, no. 1, pp. 153–184, 2010.

[50] N. Bellomo, A. Bellouquid, J. Nieto, and J. Soler, "On the multiscale modeling

of vehicular traffic: from kinetic to hydrodynamics," *Discrete and Continuous Dynamical Systems B*, vol. 19, no. 7, pp. 1869–1888, 2014.

[51] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[52] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, pp. 422–440, 2021.

[53] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[54] D. G. Patsatzis, L. Russo, I. G. Kevrekidis, and C. Siettos, "Data-driven control of agent-based models: An equation/variable-free machine learning approach," *Journal of Computational Physics*, p. 111953, 2023.

[55] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[56] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[58] A. Alahi, K. Goel *et al.*, "Social lstm: Human trajectory prediction in crowded spaces," in *CVPR*, 2016, pp. 961–971.

[59] Y. Yu, W. Xiang, and X. Jin, "Multi-level crowd simulation using social lstm," *Computer Animation and Virtual Worlds*, vol. 34, no. 3-4, p. 2180, 2023.

[60] X. Song, D. Han, J. Sun, and Z. Zhang, "A data-driven neural network approach to simulate pedestrian movement," *Physica A: Statistical Mechanics and its Applications*, vol. 509, pp. 827–844, 2018.

[61] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[62] K. Minartz, F. Hendriks, S. M. Koop, A. Corbetta, and V. Menkovski, "Understanding complex crowd dynamics with generative neural simulators," *arXiv*

preprint arXiv:2412.01491, 2024.

[63] N. Nikhil and B. T. Morris, "Convolutional neural network for trajectory prediction," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 186–196.

[64] G. Tripathi, K. Singh, and D. K. Vishwakarma, "Convolutional neural networks for crowd behaviour analysis: a survey," *The Visual Computer*, vol. 35, pp. 753–776, 2019.

[65] S. Zamboni, Z. T. Kefato, S. Girdzijauskas, C. Norén, and L. Dal Col, "Pedestrian trajectory prediction with convolutional neural networks," *Pattern Recognition*, vol. 121, pp. 108–252, 2022.

[66] Z. Yao, G. Zhang, D. Lu, and H. Liu, "Data-driven crowd evacuation: A reinforcement learning method," *Neurocomputing*, vol. 366, pp. 314–327, 2019.

[67] K. Yoshida and W. H. Warren, "Visual influence networks in walking crowds," *Journal of Vision*, vol. 23, no. 9, pp. 5175–5175, 2023.

[68] S. Huang, R. Wei, L. Lian, S. Lo, and S. Lu, "Review of the application of neural network approaches in pedestrian dynamics studies," *Heliyon*, vol. 10, no. 10, 2024.

[69] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *PNAS*, vol. 116, no. 45, pp. 22 445–22 451, 2019.

[70] A. Pecile *et al.*, "Data-driven discovery of delay differential equations with sindy," *Journal of Computational and Applied Mathematics*, 2025.

[71] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis, "Physics-informed deep neural operator networks," in *Machine Learning in Modeling and Simulation: Methods and Applications*. Springer, 2023, pp. 219–254.

[72] G. Fabiani, H. Vandecasteele, S. Goswami, C. Siettos, and I. G. Kevrekidis, "Enabling local neural operators to perform equation-free system-level analysis," *arXiv preprint arXiv:2505.02308*, 2025.

[73] S. Lee, M. Kooshkbaghi, K. Spiliotis, C. I. Siettos, and I. G. Kevrekidis, "Coarse-scale pdes from fine-scale observations via machine learning," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 1, p. 013141, 2020.

[74] E. Galaris, G. Fabiani, I. Gallos, I. Kevrekidis, and C. Siettos, "Numerical bifurcation analysis of pdes from lattice boltzmann model simulations: A parsimonious machine learning approach," *Journal of Scientific Computing*, vol. 92, no. 2, pp.

1–30, 2022.

[75] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020.

[76] S. Lee, Y. M. Psarellis, C. I. Siettos, and I. G. Kevrekidis, "Learning black- and gray-box chemotactic pdes/closures from agent based monte carlo simulation data," *Journal of Mathematical Biology*, vol. 87, no. 1, p. 15, 2023.

[77] Y. Guo, H. Zou, F. Wei, Q. Li, D. Guo, and J. Pirov, "Analysis of pedestrian second crossing behavior based on physics-informed neural networks," *Scientific Reports*, vol. 14, no. 1, pp. 21–278, 2024.

[78] G. Zhang, Z. Yu, D. Jin, and Y. Li, "Physics-infused machine learning for crowd simulation," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2439–2449.

[79] K. Luo *et al.*, "Physics-informed neural networks for pde problems: a comprehensive review," *Artificial Intelligence Review*, 2025.

[80] B. R. Zhao *et al.*, "Physics-informed neural networks for solving inverse problems: a review," *Neural Networks*, 2025.

[81] L. Lu, P. Jin, and G. E. Karniadakis, "Deeponet: Learning nonlinear operators via the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, pp. 218–229, 2021.

[82] L. Lu, X. Meng, S. Cai *et al.*, "A comprehensive and fair comparison of two neural operators (with practical extensions)," *Comput. Methods Appl. Mech. Eng.*, vol. 393, p. 114778, 2022.

[83] J. S. North *et al.*, "A review of data-driven discovery for dynamic systems," *arXiv:2210.10663*, 2022.

[84] J. N. Kutz, S. L. Brunton *et al.*, "Introduction to focus issue: Data-driven models and analysis of complex systems," *Chaos*, vol. 35, no. 3, p. 030401, 2025.

[85] A. Kirsch, "Inverse problems," in *An Introduction to the Mathematical Theory of Inverse Problems*. Springer, 2011.

[86] "Inverse problem," Wikipedia, 2025, accessed Oct. 29, 2025.

[87] P. Kothari *et al.*, "Human trajectory forecasting in crowds: A deep learning perspective," *IEEE TPAMI*, vol. 43, no. 11, pp. 3860–3877, 2021.

[88] Z. Pei *et al.*, "Human trajectory prediction in crowded scene using social-lstm," *Pattern Recognition*, vol. 93, pp. 88–100, 2019.

[89] B. Li *et al.*, "Approaches on crowd counting and density estimation: A survey," *Pattern Analysis and Applications*, vol. 24, pp. 1243–1263, 2021.

[90] M. A. Khan *et al.*, "Revisiting crowd counting: State-of-the-art, trends, and future perspectives," *arXiv:2209.07271*, 2022.

[91] G. Gao *et al.*, "A survey of deep learning methods for density estimation and crowd counting," *Intelligent Computing*, 2025.

[92] R. L. Hughes, "A continuum theory for the flow of pedestrians," *Transportation Research Part B: Methodological*, vol. 36, no. 6, pp. 507–535, 2002.

[93] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 1160–1168, 2006.

[94] A. Corbetta, A. Muntean, and K. Vafayi, "Parameter estimation of social forces in pedestrian dynamics via a probabilistic method," *Mathematical Biosciences and Engineering*, vol. 12, no. 2, pp. 337–356, 2014.

[95] N. W. F. Bode, "Parameter calibration in crowd simulation models using approximate bayesian computation," *Collective Dynamics*, vol. 5, p. A68, 2020.

[96] M. Gödel *et al.*, "Bayesian inference methods to calibrate crowd dynamics models for safety applications," *Safety Science*, vol. 149, p. 105676, 2022.

[97] D. Kim, D. Labate, K. N. Mily, and A. Quaini, "Data-driven learning to enhance a kinetic model of distressed crowd dynamics," *Math. Models Methods Appl. Sci.*, 2025, early access, 2024 preprint.

[98] C. W. Gear, J. M. Hyman, P. G. Kevrekidis, I. G. Kevrekidis, O. Runborg, and C. Theodoropoulos, "Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis," *Communications in Mathematical Sciences*, vol. 1, pp. 715–762, 2003.

[99] O. Corradi, P. G. Hjorth, and J. Starke, "Equation-free detection and continuation of a hopf bifurcation point in a particle model of pedestrian flow," *SIAM Journal on Applied Dynamical Systems*, vol. 11, no. 3, pp. 1007–1032, 2012.

[100] C. Marschler, J. Sieber, P. G. Hjorth, and J. Starke, "Equation-free analysis of macroscopic behavior in traffic and pedestrian flow," in *Traffic and Granular Flow'13*. Springer, 2014, pp. 423–439.

[101] D. G. Patsatzis, L. Russo, I. G. Kevrekidis, and C. Siettos, "Data-driven control of agent-based models: An equation/variable-free machine learning approach,"

Journal of Computational Physics, vol. 478, p. 111953, 2023.

[102] I. Panagiotopoulos, J. Starke, J. Sieber, and W. Just, "Continuation with non-invasive control schemes: Revealing unstable states in a pedestrian evacuation scenario," *SIAM Journal on Applied Dynamical Systems*, vol. 22, no. 1, pp. 1–36, 2023.

[103] T. Chin, J. Ruth, C. Sanford, R. Santorella, P. Carter, and B. Sandstede, "Enabling equation-free modeling via diffusion maps," *Journal of Dynamic Differential Equations*, vol. 36, no. Suppl 1, pp. 415–434, 2024.

[104] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Numerical gaussian processes for time-dependent and nonlinear partial differential equations," *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. A172–A198, 2018.

[105] G. Pang, L. Lu, and G. E. Karniadakis, "fpinns: Fractional physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2603–A2626, 2019.

[106] N. Ahmadi Daryakenari, M. De Florio, K. Shukla, and G. E. Karniadakis, "Ai-aristotle: A physics-informed framework for systems biology gray-box identification," *PLOS Computational Biology*, vol. 20, no. 3, p. 1011916, 2024.

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

[108] X. Li, W. Wang, Q. Liu *et al.*, "Deep neural networks for simulating complex dynamical systems: A framework combining neural operators and recurrent neural networks," *arXiv preprint arXiv:2307.07331*, 2023.

[109] K. Azizzadenesheli, N. Kovachki, Z. Li, M. Liu-Schiaffini, J. Kossaifi, and A. Anandkumar, "Neural operators for accelerating scientific simulations and design," *Nature Reviews Physics*, vol. 6, no. 5, pp. 320–328, 2024.

[110] E. Zappala, A. H. d. O. Fonseca, J. O. Caro, A. H. Moberly, M. J. Higley, J. Cardin, and D. v. Dijk, "Learning integral operators via neural integral equations," *Nature Machine Intelligence*, vol. 6, no. 9, pp. 1046–1062, 2024.

[111] A. Peyvan, V. Oommen, A. D. Jagtap, and G. E. Karniadakis, "Riemannonets: Interpretable neural operators for riemann problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 426, p. 116996, 2024.

[112] D. G. Patsatzis, M. di Bernardo, L. Russo, and C. Siettos, "Gorinns: Godunov-riemann informed neural networks for learning hyperbolic conservation laws," *Journal of Computational Physics*, vol. 534, p. 114002, 2025.

[113] A. Della Pia, D. G. Patsatzis, L. Russo, and C. Siettos, "Learning the latent dynamics of fluid flows from high-fidelity numerical simulations using parsimonious diffusion maps," *Physics of Fluids*, vol. 36, no. 10, 2024.

[114] P. G. Papaioannou, R. Talmon, I. G. Kevrekidis, and C. Siettos, "Time-series forecasting using manifold learning, radial basis function interpolation, and geometric harmonics," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 8, p. 083113, 2022.

[115] I. K. Gallos, D. Lehmberg, F. Dietrich, and C. Siettos, "Data-driven modelling of brain activity using neural networks, diffusion maps, and the koopman operator," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 34, no. 1, 2024.

[116] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*. Springer, 2006, pp. 366–381.

[117] D. Helbing, "Traffic and related self-driven many-particle systems," *Reviews of Modern Physics*, vol. 73, no. 4, pp. 1067–1141, 2001.

[118] P. Ball, *Critical Mass: How One Thing Leads to Another*. Farrar, Straus and Giroux, 2004.

[119] D. Helbing and A. Johansson, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, 2000.

[120] G. K. Still, "Introduction to crowd science," *CRC Press*, 2014.

[121] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos, "Equation-free multiscale computation," *Communications in Mathematical Sciences*, vol. 1, no. 4, pp. 715–762, 2004.

[122] M. Scheffer *et al.*, "Early-warning signals for critical transitions," *Nature*, vol. 461, pp. 53–59, 2009.

[123] H. V. Alvarez, D. G. Patsatzis, L. Russo, I. G. Kevrekidis, and C. Siettos, "Next generation equation-free multiscale modelling of crowd dynamics via machine learning," *Journal of Computational Physics*, vol. 559, p. 114881, 2026.

[124] H. Vargas Alvarez, G. Fabiani, N. Kazantzis, I. G. Kevrekidis, and C. Siettos, "Nonlinear discrete-time observers with physics-informed neural networks," *Chaos, Solitons & Fractals*, vol. 186, p. 115215, 2024.

[125] H. V. Alvarez, G. Fabiani, N. Kazantzis, C. Siettos, and I. G. Kevrekidis, "Discrete-time nonlinear feedback linearization via physics-informed machine learning," *Journal of Computational Physics*, vol. 492, p. 12408, 2023.

[126] W. E, *Multiscale Modeling and Simulation*. Berlin, Germany: Springer, 2011.

[127] W. E and B. Engquist, *Principles of Multiscale Modeling*. Cambridge, UK: Cambridge University Press, 2003.

[128] A. J. Chorin and O. H. Hald, *Stochastic Tools in Mathematics and Science*. New York, NY, USA: Springer, 2006.

[129] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and K. Theodoropoulos, *Equation-Free Multiscale Computation: Enabling Microscopic Simulators to Perform System-Level Analysis*. Princeton, NJ, USA: Princeton University Press, 2009.

[130] C. Marschler, J. Sieber, R. Berkemer, A. Kawamoto, and J. Starke, "Implicit methods for equation-free analysis: Convergence results and analysis of emergent waves in microscopic traffic models," *SIAM Journal on Applied Dynamical Systems*, vol. 13, no. 3, pp. 1202–1238, 2014.

[131] C. I. Siettos, I. G. Kevrekidis, and D. Maroudas, "Coarse bifurcation diagrams via microscopic simulators: a state-feedback control-based approach," *International Journal of Bifurcation and Chaos*, vol. 14, no. 01, pp. 207–220, 2004.

[132] I. G. Kevrekidis, C. W. Gear, and G. Hummer, "Equation-free: The computer-aided analysis of complex multiscale systems," *AIChE Journal*, vol. 49, no. 6, pp. 1346–1355, 2003.

[133] C. I. Siettos and I. G. Kevrekidis, "Equation-free multiscale computation: Enabling microscopic simulators to perform system-level analysis," *Chemical Engineering Science*, vol. 59, no. 13, pp. 2469–2481, 2004.

[134] A. Quarteroni, G. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations*. Cham, Switzerland: Springer, 2016.

[135] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 2012.

[136] R. Temam, *Navier–Stokes equations and nonlinear functional analysis*. SIAM, 1995.

[137] ——, *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*, 2nd ed., ser. Applied Mathematical Sciences. Springer, 1997, vol. 68.

[138] L. Sirovich, "Turbulence and the dynamics of coherent structures. I-III," *Quarterly of Applied Mathematics*, vol. 45, no. 3, pp. 561–590, 1987.

[139] C. W. Rowley, "Model reduction for fluids using balanced proper orthogonal

decomposition," *International Journal of Bifurcation and Chaos*, vol. 15, no. 3, pp. 997–1013, 2005.

[140] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[141] P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge, UK: Cambridge University Press, 1996.

[142] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, *Model Reduction and Approximation*. Philadelphia, PA, USA: SIAM, 2017.

[143] J. N. Kutz, *Data-Driven Modeling & Scientific Computation*. Oxford, UK: Oxford University Press, 2013.

[144] S. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[145] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[146] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[147] K.-I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous-time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[148] P. Vlachas, J. Pathak, B. R. Hunt, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," *Neural Networks*, vol. 126, pp. 191–217, 2020.

[149] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1310–1318, 2013.

[150] J. D. Hamilton, *Time Series Analysis*. Princeton, NJ, USA: Princeton University Press, 1994.

[151] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Berlin, Germany: Springer, 2005.

[152] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia, PA,

USA: SIAM, 2016.

[153] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[154] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[155] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Springer, 2005.

[156] N. Bellomo, B. Piccoli, and A. Tosin, "Modeling crowd dynamics from a complex systems viewpoint," *Mathematical Models and Methods in Applied Sciences*, vol. 22, no. suppl., p. 1230004, 2012.

[157] S. T. Rassia and C. I. Siettos, "Escape dynamics in office buildings: using molecular dynamics to quantify the impact of certain aspects of human behavior during emergency evacuation," *Environmental Modeling & Assessment*, vol. 15, pp. 411–418, 2010.

[158] N. Bellomo and L. Gibelli, "Toward a mathematical theory of behavioral-social dynamics for pedestrian crowds," *Mathematical Models and Methods in Applied Sciences*, vol. 25, no. 13, pp. 2417–2437, 2015.

[159] N. Bellomo, D. Clarke, L. Gibelli, P. Townsend, and B. Vreugdenhil, "Human behaviours in evacuation crowd dynamics: From modelling to "big data" toward crisis management," *Physics of Life Reviews*, vol. 18, pp. 1–21, 2016.

[160] M. Haghani, E. Cristiani, N. W. Bode, M. Boltes, and A. Corbetta, "Panic, irrationality, and herding: three ambiguous terms in crowd dynamics research," *Journal of Advanced Transportation*, vol. 2019, no. 1, p. 9267643, 2019.

[161] N. Bellomo, S.-Y. Ha, and N. Outada, "Towards a mathematical theory of behavioral swarms," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 26, p. 125, 2020.

[162] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, May 1995.

[163] F. Golse, "On the dynamics of large particle systems in the mean field limit," *Lecture Notes*, 2016.

[164] L. Devroye, "The equivalence of weak, strong and complete convergence in l1 for kernel density estimates," *The Annals of Statistics*, pp. 896–904, 1983.

[165] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, ser. Monographs on Statistics and Applied Probability. London: Chapman and Hall, 1986.

[166] V. Morand, N. Mueller, R. Weightman, B. Piccoli *et al.*, "Deep learning of first-order nonlinear hyperbolic conservation law solvers," *arXiv preprint arXiv:2307.04121*, 2023.

[167] N. Aubry, W.-Y. Lian, and E. S. Titi, "Preserving symmetries in the proper orthogonal decomposition," *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 483–505, 1993.

[168] T. Sauer, J. A. Yorke, and M. Casdagli, "Embedology," *Journal of Statistical Physics*, vol. 65, no. 3–4, pp. 579–616, 1991.

[169] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[170] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American Statistical Association*, vol. 74, no. 366a, pp. 427–431, 1979.

[171] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[172] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[173] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American Statistical Association*, vol. 74, no. 366, pp. 427–431, 1979.

[174] The MathWorks Inc., "Matlab," Natick, Massachusetts, United States, 2024.

[175] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[176] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1171–1179.

[177] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[178] F. P. Kemeth, T. Bertalan, N. Evangelou, T. Cui, S. Malani, and I. G. Kevrekidis, "Initializing lstm internal states via manifold learning," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 9, 2021.

[179] M. Benosman, H. Mansour, and V. Huroyan, "Koopman-operator observer-based

estimation of pedestrian crowd flows," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 028–14 033, 2017, 20th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896317332949

[180] S. T. Chung and J. W. Grizzle, "Sampled-data observer error linearization," *Automatica*, vol. 26, p. 997, 1990.

[181] W. Lee and K. Nam, "Observer design for autonomous discrete-time nonlinear systems," *Systems and Control Letters*, vol. 17, p. 49, 1991.

[182] W. Lin and C. I. Byrnes, "Remarks on linearization of discrete-time autonomous systems and nonlinear observer design," *Systems and Control Letters*, vol. 25, p. 31, 1995.

[183] N. Kazantzis and C. Kravaris, "Discrete-time observer design using functional equations," *Systems and Control Letters*, vol. 42, pp. 81–94, 2001.

[184] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[185] M. Xiao, N. Kazantzis, C. Kravaris, and A. J. Krener, "Nonlinear discrete-time observer design with linearizable error dynamics," *IEEE Transactions on Automatic Control*, vol. 48, pp. 622–626, 2003.

[186] H. V. Alvarez, G. Fabiani, N. Kazantzis, C. Siettos, and I. G. Kevrekidis, "Discrete-time nonlinear feedback linearization via physics-informed machine learning," *Journal of Computational Physics*, vol. 492, p. 12408, 2023.

[187] A. Isidori, *Nonlinear Control Systems*, ser. Communications and Control Engineering. Springer London, 1995.

[188] D. Luenberger, "Observing the state of a linear system," *IEEE Transactions on Military Electronics*, vol. 8, no. 2, pp. 74–80, 1963.

[189] N. Kazantzis, "A functional equations approach to nonlinear discrete-time feedback stabilization through pole-placement," *Systems and Control Letters*, vol. 43, no. 5, pp. 361–369, 2001.

[190] L. B. Rall, "Automatic differentiation: Techniques and applications," in *Lecture Notes in Computer Science*, 1981.

[191] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks." *Journal of machine learning research*, vol. 10, no. 1, 2009.

[192] G. Fabiani, E. Galaris, L. Russo, and C. Siettos, "Parsimonious physics-informed random projection neural networks for initial-value problems of odes and index-1 daes," *arXiv preprint arXiv:2203.05337*, 2022.

[193] P. G. Kevrekidis, C. Siettos, and Y. G. Kevrekidis, "To infinity and some glimpses of beyond," *Nature communications*, vol. 8, no. 1, p. 1562, 2017.

[194] M. E. Henderson, "Computing invariant manifolds by integrating fat trajectories," *SIAM Journal on Applied Dynamical Systems*, vol. 4, no. 4, pp. 832–882, 2005.

[195] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, p. 047510, 2012.

[196] E. M. Bollt, Q. Li, F. Dietrich, and I. Kevrekidis, "On matching, and even rectifying, dynamical systems through koopman operator eigenfunctions," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 2, pp. 1925–1960, 2018.

[197] G. Fabiani, F. Calabrò, L. Russo, and C. Siettos, "Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines," *Journal of Scientific Computing*, vol. 89, pp. 1–35, 2021.

[198] E. Galaris, G. Fabiani, I. Gallos, I. Kevrekidis, and C. Siettos, "Numerical bifurcation analysis of pdes from lattice boltzmann model simulations: a parsimonious machine learning approach," *Journal of Scientific Computing*, vol. 92, no. 2, p. 34, 2022.

[199] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. part i: Theory," *AICHE Journal*, vol. 65, pp. 1–14, 2019.

[200] Z. Wu, D. Rincon, and P. D. Christofides, "Real-time adaptive machine-learning-based predictive control of nonlinear processes," *Industrial and Engineering Chemistry Research*, vol. 59, no. 6, pp. 2275–2290, 2020.

[201] Z. Wu, D. Rincon, Q. Gu, and P. D. Christofides, "Statistical machine learning in model predictive control of nonlinear processes," *Mathematics*, vol. 9, p. 1912, 2021.

[202] Z. Wu, A. Alnajd, Q. Gu, and P. D. Christofides, "Statistical machine-learning-based predictive control of uncertain nonlinear processes," *AICHE Journal*, vol. 68, p. 17642, 2022.

[203] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis, "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons," *Journal of Computational Physics*, vol. 477, p. 111902, 2023.

[204] M. P. Wand and M. C. Jones, *Kernel Smoothing*. Chapman and Hall, 1995.

[205] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability and Its Applications*, vol. 14, no. 1, pp. 153–158, 1969.

[206] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[207] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[208] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969.

[209] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[210] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[211] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, 1993.

[212] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[213] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[214] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012, pp. 1097–1105.

[215] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[216] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[217] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[218] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014.

[219] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[220] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[221] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993.

[222] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[223] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *The Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[224] F. Calabrò, G. Fabiani, and C. Siettos, "Extreme learning machine collocation for the numerical solution of elliptic pdes with sharp gradients," *Computer Methods in Applied Mechanics and Engineering*, vol. 387, p. 114188, 2021.

[225] M. De Florio, E. Schiassi, and R. Furfaro, "Physics-informed neural networks and functional interpolation for stiff chemical kinetics," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 6, 2022.

[226] G. Fabiani, E. Galaris, L. Russo, and C. Siettos, "Parsimonious physics-informed random projection neural networks for initial value problems of odes and index-1 daes," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 4, 2023.

[227] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: a deep learning library for solving differential equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, 2021.

[228] S. Dong and Y. Wang, "A method for computing inverse parametric pde problems with random-weight neural networks," *Journal of Computational Physics*, vol. 489, p. 112263, 2023.

[229] Y. Qian, Y. Zhang, Y. Huang, and S. Dong, "Physics-informed neural networks for approximating dynamic (hyperbolic) pdes of second order in time: Error analysis and algorithms," *Journal of Computational Physics*, vol. 495, p. 112527, 2023.

[230] E. Schiassi, M. De Florio, A. D'Ambrosio, D. Mortari, and R. Furfaro, "Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models," *Mathematics*, vol. 9, no. 17, p. 2069, 2021.

[231] F. Dietrich, A. Makeev, G. Kevrekidis, N. Evangelou, T. Bertalan, S. Reich, and I. G. Kevrekidis, "Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 2, p. 023121, 2023.

[232] I. Kičić, P. R. Vlachas, G. Arampatzis, M. Chatzimanolakis, L. Guibas, and P. Koumoutsakos, "Adaptive learning of effective dynamics for online modeling of complex systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 415, p. 116204, 2023.

[233] B. Champenois and T. Sapsis, "Machine learning framework for the real-time reconstruction of regional 4d ocean temperature fields from historical reanalysis data and real-time satellite and buoy surface measurements," *Physica D: Nonlinear Phenomena*, vol. 459, p. 134026, 2024.

[234] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces with applications to pdes," *Journal of Machine Learning Research*, vol. 24, no. 89, pp. 1–97, 2023.

[235] P. Karnakov, S. Litvinov, and P. Koumoutsakos, "Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks," *PNAS nexus*, vol. 3, no. 1, p. pgae005, 2024.

[236] N. A. Daryakenari, M. De Florio, K. Shukla, and G. E. Karniadakis, "Ai-aristotle: A physics-informed framework for systems biology gray-box identification," *PLOS Computational Biology*, vol. 20, no. 3, 2024.

[237] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature machine intelligence*, vol. 3, no. 3, pp. 218–229, 2021.

[238] R. Tipireddy, P. Perdikaris, P. Stinis, and A. M. Tartakovsky, "Multistep and con-

tinuous physics-informed neural network methods for learning governing equations and constitutive relations," *Journal of Machine Learning for Modeling and Computing*, vol. 3, no. 2, 2022.

[239] D. G. Patsatzis, G. Fabiani, L. Russo, and C. Siettos, "Slow invariant manifolds of singularly perturbed systems via physics-informed machine learning," *arXiv preprint arXiv:2309.07946*, 2023.