Deep Learning for Reduced Order Modeling

Emmanuel Menier

université PARIS-SACLAY

Deep Learning for Reduced Order Modeling

Apprentissage Machine pour la Réduction de Modèles

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique mathématique
Graduate School : Informatique et sciences du numérique
Référent : Faculté des sciences d'Orsay

Thèse préparée dans les unités de recherche LISN (Université Paris-Saclay, CNRS) et Inria Saclay-Ile-de-France (Université Paris-Saclay, Inria), sous la direction de **Marc SCHOENAUER**, directeur de recherche INRIA, le co-encadrement de **Michele Alessandro BUCCI** ingénieur de recherche à Safran Tech et **Mouadh YAGOUBI**, responsable de projet R&D à IRT-SystemX.

Thèse soutenue à Paris-Saclay, le 25 janvier 2024, par

Emmanuel MENIER

Composition du jury

Membres du jury avec voix délibérative

Taraneh SAYADI Professeur, CNAM, Paris	Examinatrice & Presidente du jury
Gianluigi ROZZA Professeur, SISSA, Trieste	Rapporteur & Examinateur
Amaury HABRARD Professeur, Université Jean Monnet de Saint-Etienne	Rapporteur & Examinateur
Nicolas THOME Professeur, Sorbonne Université, Paris	Examinateur
Phaedon-Stelios KOUTSOURELAKIS Professor, Technical University Munich	Examinateur
Johannes BRANDSTETTER Professeur Assistant, JKU, Linz	Examinateur

THESE DE DOCTORAT
NNT : 2024UPASG004

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage Machine pour la Réduction de Modèles
Mots clés: Apprentissage Machine, Modèles réduits, Séries temporelles, Systèmes dynamiques

Résumé: Les systèmes dynamiques sont généralement modélisés à l'aide d'équations aux dérivées partielles (EDP). Ces modèles sont étroitement liés à la façon dont les scientifiques observent le monde et, en tant que tels, ils sont limités par notre compréhension des systèmes étudiés. En effet, des modèles tels que les équations de Navier-Stokes ne modélisent que les interactions locales dans un écoulement, et négligent les phénomènes sous-jacents qui contrôlent le système dans son ensemble. Cela conduit souvent à des coûts de calcul excessifs associés à la résolution numérique des EDP. Dans cette thèse, nous discutons de la manière dont les données dynamiques peuvent être exploitées pour dériver de meilleurs espaces de représentation pour les systèmes physiques ainsi que des modèles simplifiés, appelés modèles réduits. Nous présentons d'abord quelques unes des approches de réduction de modèle existantes. Nous proposons ensuite d'exploiter les capacités d'approximation des ***réseaux de neurones*** pour construire de nouvelles méthodes de réduction de modèles. Les techniques introduites dans cette thèse reposent sur le concept d'***hybridation*** entre la modélisation physique et les méthodes d'apprentissage machine. Nous nous appuyons sur les propriétés des systèmes dynamiques étudiés pour construire des modèles interprétables, précis et en accord avec la théorie afin de résoudre les problèmes de coûts de calcul associés à la modélisation physique standard, tout en limitant la dépendance des modèles aux données. Nous proposons deux nouvelles approches, la méthode ***CD-ROM*** qui propose de construire des modèles de fermeture les modèles réduits par la méthode POD-Galerkin, et la méthode ***iLED***, qui est une approche de modélisation entièrement basée sur les données, construisant des modèles dynamiques interprétables à l'aide de réseaux de neurones. Chaque méthode est illustrée par des expériences numériques sur des cas tests standards tirés de la littérature comme les écoulements en deux dimensions, ou les systèmes chaotiques tels que l'équation de Kuramoto-Sivashinsky.

Title: Deep Learning for Reduced Order Modeling
Keywords: Deep Learning, Reduced Order Modeling, Time series, Dynamical systems

Abstract: Dynamical systems are generally modeled using Partial Differential Equations (PDE). These models are intricately linked to the way scientists observe the world and, as such, they are limited by our understanding of the behavior of the systems under study. For example, models such as the Navier-Stokes equations only account for the local interactions in fluid systems, and ignore the underlying phenomena that drive the system as a whole. This often leads to a poor understanding of the dynamical problems under study and excessive computational costs associated with the numerical resolution of PDE-based models. In this thesis, we discuss the way dynamical data can be exploited to derive better representation spaces for physical systems as well as computationally efficient models, called reduced order models. We discuss some of the existing reduced order modeling approaches. We then propose to leverage the approximation power of ***neural networks*** to derive novel, improved reduced order modeling methods. The modeling techniques proposed in this thesis are built around the concept of ***hybridization*** between physical and data driven modeling. We leverage pre-existing knowledge of dynamical systems into theoretically grounded, accurate, and interpretable dynamical models to address the computational costs issues associated with standard physical modeling, while avoiding complete reliance on data. We introduce two novel modeling approaches, the ***CD-ROM*** method which proposes to construct neural closure models for the well established POD-Galerkin method, and the ***iLED*** method, which is a fully data driven modeling approach that extracts low dimensional, interpretable dynamical models from data using neural networks. Each method is illustrated using standard cases from the literature such as two dimensional fluid flows, and chaotic systems such as the Kuramoto-Sivashinsky equations.

4

REMERCIEMENTS

En premier lieu, je souhaite remercier mes encadrants pour leur soutien et leur aide durant cette thèse. Merci à Alessandro pour nos discussions quotidiennes et toutes tes idées qui sont à l'origine d'une bonne partie de ces travaux. Merci à Lionel d'avoir accepté de rejoindre l'équipe d'encadrement en cours de thèse, et pour l'aide que tu as pu m'apporter. Merci à Mouadh et Marc pour leur confiance, leurs conseils, leur soutien et la liberté qu'ils m'ont accordé durant cette thèse.

Merci aux rapporteurs de cette thèse, Amaury Habrard et Gianluigi Rozza pour leurs retours détaillés sur ce manuscript. Merci aussi aux examinateurs, Nicolas Thome, Phaedon-Stelios Koutsourelakis, Taraneh Sayadi et Johannes Brandstetter de m'avoir fait l'honneur de participer à mon jury.

Merci à Petros Koumoutsakos de m'avoir accueilli à Boston, et Sebastian pour sa collaboration et ses conseils.

Merci à Matthieu et Lucas pour l'entraide et les échanges que nous avons pu avoir dans notre bureau, au café et autour d'une bière. Merci aussi à Thibault, Alice, Amine, Manon, Rémy et tous les chercheurs et doctorants des équipes TAU et Decipher qui ont participé à créer une ambiance conviviale au laboratoire.

Merci à tous mes amis, et particulièrement Lucas, pour ces moments d'évasion, partagés loin du travail. Merci aussi à ma famille, mes parents, mes frères et ma soeur.

Pour finir, merci à Océanne, merci de m'avoir suivi à Paris. Merci de toujours m'encourager à faire ce que j'aime, et me soutenir quand je m'en plains. C'est à toi que je dois d'avoir tant apprécié ces dernières années.

CONTENTS

1 Introduction **1**

2 Reduced Order Modeling **7**
2.1 Full order simulation 7
2.1.1 First principles 8
2.1.2 Problem Discretisation 9
2.1.3 Accuracy & Computational Cost 11
2.2 Dimensionality Reduction 13
2.2.1 Intrinsic Dimension 13
2.2.2 Linear reduction 14
2.2.3 Non Linear reduction 19
2.3 Conclusion 23

3 Neural Networks for dynamical modeling **25**
3.1 Introduction to Deep Learning 26
3.1.1 The Multi Layer Perceptron 27
3.1.2 Backpropagation 27
3.1.3 Stochastic Gradient Descent 30
3.1.4 Inductive Bias 32
3.1.5 Neural networks for dimensionality reduction 33
3.2 Learning the Dynamics 34
3.2.1 Fully data-driven models 36
3.2.2 On the importance of Hybridisation 41
3.2.3 Deep Learning and physics 42
3.2.4 Summary 49
3.3 The POD Galerkin method 49
3.4 Conclusion 53

4 CD-ROM: Complementary Deep - Reduced Order Model 55
4.1 Introduction . . . 56
4.1.1 Context . . . 56
4.1.2 Related Work . . . 57
4.2 Model reduction approach . . . 59
4.2.1 POD Galerkin . . . 59
4.2.2 Non-Markovianity and Takens' theorem . . . 60
4.3 Data driven learning of the residual . . . 63
4.3.1 Neural Networks . . . 63
4.3.2 Memory time scales . . . 63
4.3.3 Training strategy . . . 64
4.3.4 Implementation details . . . 65
4.4 Interpretation of the model . . . 68
4.4.1 The Mori-Zwanzig formalism . . . 68
4.4.2 Deep Learning interpretation of the model . . . 69
4.5 Case presentation and reduced models . . . 71
4.5.1 Flow over a cylinder . . . 71
4.5.2 Fluidic pinball . . . 73
4.5.3 Parametric Kuramoto-Sivashinsky equation . . . 75
4.6 Results and discussion . . . 77
4.6.1 Cylinder case . . . 78
4.6.2 Fluidic pinball results . . . 80
4.6.3 Parametric KS equation results . . . 82
4.7 Additional Results . . . 86
4.7.1 Number of modes for the fluidic pinball reduction . . . 86
4.7.2 Computational cost . . . 88
4.8 Conclusion . . . 90

5 Adaptive hybridization and Memory Interpretability 93
5.1 Introduction . . . 93
5.2 Application to Calendering and adaptive hybridization . . . 94
5.2.1 Problem introduction . . . 95
5.2.2 Proposed modeling approach . . . 96
5.2.3 Results . . . 98
5.2.4 Study Conclusion . . . 100
5.3 Memory interpretability and Time horizons . . . 102
5.3.1 Introduction . . . 102
5.3.2 Memory *Horizons* . . . 102
5.3.3 Experiment setup . . . 103
5.3.4 Modeling . . . 104
5.3.5 Results . . . 104
5.4 Conclusion . . . 106

6 iLED: interpretable Learning of Effective Dynamics for multiscale systems 107
6.1 Introduction . . . 108
6.2 Methodology . . . 109
6.2.1 Dimensionality Reduction . . . 110
6.2.2 Framing *iLED* within the Mori-Zwanzig formalism . . . 111
6.2.3 The *iLED* architecture . . . 113
6.2.4 Training the *iLED* architecure . . . 115
6.3 Implementation Details . . . 117
6.3.1 Memory Initialization . . . 118
6.3.2 Linear Parameterization . . . 118
6.3.3 Latent space centering . . . 118
6.3.4 Form of the memory kernel network . . . 119
6.4 Numerical Experiments . . . 119
6.4.1 Example 1: The FitzHugh-Nagomo Model . . . 119
6.4.2 Example 2: The Kuramoto-Sivashinsky Equation . . . 122
6.4.3 Example 3: Navier-Stokes Equations for the Flow around a Cylinder . . . 123
6.4.4 Remarks on the linearity of the dynamics . . . 127
6.5 conclusion . . . 128

7 Additional Studies: Dynamical Systems for Generative Modeling 131
7.1 Introduction . . . 131
7.2 Unsupervised Domain Translation . . . 132
7.2.1 Method . . . 133
7.2.2 Results . . . 135
7.2.3 Study Conclusion . . . 137
7.3 Hamiltonian Flows for Generative Modeling . . . 137
7.3.1 Methods . . . 139
7.3.2 Related Works . . . 144
7.3.3 Experiments . . . 146
7.3.4 Limitations and Open questions . . . 150
7.3.5 Study Conclusion . . . 152
7.4 Conclusion . . . 152

8 Conclusion 155
8.1 Perspectives . . . 156

Appendices

9 Appendix: CD-ROM 161
9.1 Hyper-Parameters and training . . . 161

10 Appendix: iLED **163**
10.1 Network parameters 163
10.1.1 FHN 163
10.1.2 KS 165
10.1.3 Flow around a cylinder 167

11 Synthèse en Français **171**

LIST OF FIGURES

2.1 Computational domain of the cylinder flow. Typical boundary conditions for this problem are also displayed. $\vec{U}_\infty$ is the inflow velocity and σ the fluid stress tensor. 9

2.2 A 1-dimensional domain meshed into five sub-domains, over which six piecewise linear basis functions are defined. 10

2.3 Vorticity field of the case of isotropic turbulence represented on meshes of various precision. Coarse meshes are unable to represent small-scale phenomena and filter out critical information. *Turbulence data obtained from Biferale et al. [127]*. 11

2.4 Mesh used for the simulation of the cylinder flow case. 12

2.5 The idea of dimensionality reduction. The solution points lie on a lower dimensional (in *red*) manifold than the full space used to represent them. 13

2.6 *Left:* Points lying on a two-dimensional manifold, embedded in a three-dimensional space. *Right:* Projection of the data on the leading two POD modes. 19

2.7 *Left:* Points lying on a two-dimensional manifold, embedded in a three- dimensional space. *Right:* Two-dimensional embedding of the data, obtained with the LLE algorithm. 21

2.8 *Left:* Points lying on a two-dimensional manifold, embedded in a three- dimensional space. *Right:* Two-dimensional embedding of the data, obtained with Isomaps. The conservation of the geodesic distance is also illustrated in *black* with the shortest path between two points on the 3D manifold presented on the left, and the equivalent Euclidean distance in the reduced space on the right. . 22

3.1 Representation of a Multi Layer Perceptron ($\Phi(x)$) with 3 hidden layers. 28

3.2 Visualization of various gradient descents algorithms. 32
3.3 Illustration of an autoencoder used to learn the map from data samples $\boldsymbol{u}$ to a low dimensional embedding z 34
3.4 Performance of a neural Autoencoder on the reduction of a three-dimensional data manifold. *Top:* Two dimensional embedding and reconstruction of the data learned by the Autoencoder. *Bottom:* Evolution of the data samples projected on the leading principal components of the encoded data at each layer. 35
3.5 Forward and backward time integration of the Van der Pol oscillator. Numerical errors during the integration lead to the divergence of the two trajectories which should be identical. 48
3.6 *Hybridization spectrum* between full order methods and fully data-driven methods. NB: The annotated methods only constitute a sample of the various approaches relevant to the thesis topic, and are chosen to illustrate the variety of possible approaches. 53

4.1 The CD-ROM method on the hybridization spectrum. 57
4.2 Illustration of the Takens' delay embedding theorem. Left: Original Lorenz attractor; middle: Delayed time series of the X coordinate; right: Reconstructed attractor. 61
4.3 Visualisation of the CD-ROM approach. The full order solution snapshots are projected on the POD basis to obtain time series of the reduced coordinates ($\boldsymbol{a}$). These reduced coordinates are then augmented with a memory $\mathbf{y}(t)$. Finally, the augmented dynamical model is presented on the right hand side of the image. . 63
4.4 Superposition of sinusoidal signals of frequencies 1, 10 and 100 Hz, filtered by an exponential decay with a decay rate $\lambda = 50$. The higher frequency (100 Hz) is filtered out while the other two are reproduced in the memory signal. This implies that recent events in memory due to the 100 Hz frequency are filtered out so that the memory is mostly driven by *older* events associated with lower frequencies. Note that the memory signal has been scaled by a factor λ for clarity. 64
4.5 Computational domain for the cylinder case. 71
4.6 Vorticity fields of the modes selected for the reduced order modeling of the cylinder flow. 72
4.7 True trajectory and simulation of an uncorrected Galerkin ROM. Plots (1) & (2) show the phase space trajectories projected on the $\boldsymbol{a}_1 - \boldsymbol{a}_\triangle$ and $\boldsymbol{a}_1 - \boldsymbol{a}_2$ planes, while (3) shows the time evolution of the shift mode coefficient. 73
4.8 Computational domain of the fluidic pinball case. 74
4.9 Flow complexity of the pinball case. Left: Vorticity field of the flow in the chaotic regime. Right: Spectrum of the snapshot matrix. . . 74

4.10 From top to bottom: time evolution of the first 3 POD coefficients. *Black line*: true value of the pinball's POD coefficients, *orange line*: trajectories obtained by integrating an uncorrected Galerkin reduced order model constructed using the first 10 POD modes. . . 75

4.11 *Left*: Simulations of the KS equation carried out under two different parameter values. *Right*: Relative snapshot reconstruction accuracy against the number of POD modes used for reconstruction. 76

4.12 Coefficients of the first 4 POD modes simulated with the uncorrected Galerkin ROM under different parameter values. *Dashed line*: Projected DNS data, *full line*: Simulation of uncorrected the Galerkin ROM. 77

4.13 CD-ROM trajectory obtained at different levels of training. The first mode's amplitude is displayed in the top panel, while the third mode's amplitude is shown in the bottom panel. 79

4.14 Results obtained with the corrected ROM for the simulation of the cylinder flow's transient dynamics. Left: phase space trajectory from base flow to limit cycle simulated with the finite element solver, the original Galerkin ROM and our corrected model. Right: time evolution of the shift mode's coefficient simulated with the same models, as well as the norm of the correction applied by our model. 79

4.15 Simulation results obtained on a test trajectory. As in Figure 4.14, the first graph represents the phase space trajectory, while the time evolution of the shift mode's coefficient is presented on the second graph. 80

4.16 Vorticity fields obtained after 110 s of simulation in the chaotic regime. 81

4.17 Results of the correction approach applied to the 10-mode ROM of the fluidic pinball. The initial condition is a random point selected in the training trajectory. Plots (a),(b),(c) describe the coefficients of the first three POD modes simulated with different models. Plot (d) presents the relative Euclidean distance between the true data and the trajectory simulated with the CD-ROM. . . . 82

4.18 Statistics of the 10-dimensional attractors described by the projected DNS data (black) and the CD-ROM (blue). Left: Average number of points g in a sphere of radius r, the data follows the law $g = r^{C_d}$ where C_d corresponds to the correlation dimension of the attractor. Right: Time evolution of the average distance d between trajectories starting from arbitrarily close initial conditions on the attractor. The data follows the law $d = e^{tl_e}$ where l_e corresponds to the maximum lyapunov exponent of the system at hand. 83

4.19 Estimated probability density functions for the coefficient of each mode. *Plain blue line*: Statistics of the trajectory simulated with the corrected model; *dotted black line*: Statistics of the projected DNS data. Labels refer to the mode index. 83

4.20 Coefficients of the first 4 POD modes simulated with the CD-ROM as well as the uncorrected Galerkin ROM under different parameter values in the test dataset. *Dashed line*: Projected DNS data, *full orange line*: Simulation of the Galerkin ROM, *full blue line*: Simulation of the CD-ROM. 84

4.21 Error metric (equation 4.37) computed for each test parameter value. 85

4.22 Comparison of the performance of POD-Galerkin models at different degrees of reduction. *Top*: Simulated value for the amplitude of the first POD mode; *center*: Value of the closure term for the first POD mode computed on the true trajectory; *bottom*: Relative distance between simulated and true trajectories. 87

4.23 Loss evolution of two CD-ROM architectures using a different number of modes. 88

4.24 Performance and computational cost of simulating different models starting from an initial condition outside the CD-ROM training basis. 89

5.1 The adaptive CD-ROM approach on the hybridization spectrum. . 94

5.2 *Left*: Schematic of the calendering process. *Right*: Simulation mesh used in the finite element solver. 95

5.3 Temperature field POD. *Top:* Relative training data reconstruction error depending on the number of modes. *Bottom:* Visualization of the leading POD modes. 98

5.4 Velocity field POD. *Top:* Relative training data reconstruction error depending on the number of modes. *Bottom:* Visualization of the vertical component of the leading POD modes. 99

5.5 Performance of the corrected ROM (Eq. (5.6)) on a trajectory unseen during training. *Left*: Trajectory of the first mode a_1. *Right*: Trajectory of the second mode a_2. The incomplete ROM ($\mathbf{r}$ in Eq. (5.5)) is also shown for comparison. 100

5.6 A single frequency sine wave results from a two-dimensional system, it can not be modeled through one-dimensional Markovian dynamics. However, a delayed coordinate can be used to model the dynamics as a non-markovian process. 103

5.7 True and predicted values obtained after training the CD-ROM architecture in Eq.(5.15) with three different initial values τ_0 for the time horizons. 105

5.8 Distributions of the learned time horizons at the end of the training, starting from 50 constant initializations. Average values for the time horizons are denoted as $< \tau_i >$. Three cases did not lead to a converged model and are not presented here. 105

6.1 *iLED* on the hybridization spectrum. 109

6.2 ILED architecture: The high-dimensional system is encoded to a lower-dimensional representation using the encoder $\mathcal{E}$. The lower-dimensional representation is propagated in time using a linear and a non-linear part based on the Mori-Zwanzig formalism. With the help of a decoder $\mathcal{D}$, the high-dimensional system is subsequently reconstructed. 114

6.3 Visualization of the FHN model's dynamics. The evolution of the full state for a subset of the test trajectory is presented on the left. The right hand side of the plot displays the latent manifold learned by an autoencoder using latent dimension $d_z = 2$. 120

6.4 Forecasting performance of the *iLED* method on the FHN case. From top to bottom: true inhibitor field, predicted inhibitor field and absolute error between the two. The right hand side presents the true and predicted latent trajectory for an integration period of $8000s$. *NB:* only the inhibitor v field is presented for clarity, as it is harder to predict than the activator field u. 121

6.5 Norm of the dynamics parts 121

6.6 Two views of a training trajectory for the Kuramoto-Shivasinsky case (see text). 123

6.7 Results obtained with the *iLED* method on a test trajectory. *Dashed black line*: Horizon of the warm-up required to initialize the memory of the model, *Dashed red lined*: time horizon used to train the model. 124

6.8 *Top*: Eigenvalues λ_i of the *iLED* linear operator (average and std. dev. over ten different training runs). *Bottom*: Fourier transform of a test trajectory averaged over the computational domain, the natural periods of the *iLED* operator are also displayed for comparison. Note that several runs learned one purely real eigenvalue, meaning that the learned period is infinite, thus not included in the computation of the largest period (T_4). 125
6.9 Multiscale architecture used to model the cylinder flow. The area around the cylinder is rendered at four times the resolution of the rest of the field, as it is where the dynamics are most complex. . . 126
6.10 Results obtained with the *iLED* method on the case of the cylinder flow at a Reynolds number of 100. 127
6.11 Results obtained with the iLED method on the case of the cylinder flow at a Reynolds number of 750. 128

7.1 Selected samples of the male to female transport process using the proposed continuous domain translation approach. The transported encodings are decoded at regular time intervals, to illustrate the transformation applied by the model. 135
7.2 Selected samples of the reverse male to female generation process. NB : These samples are generated using equation (7.3) as transport flow was solely trained to map male to females. 136
7.3 Results of excessive integration of the transport flow. A model trained to map males to females in 1 $t.u.$ is integrated backward for more than twice the map horizon. We observe that the generated samples retain semantic sense for up to about 1.5 times the training horizon. 136
7.4 Illustration of the invertibility of conservative systems. The ideal (conservative) pendulum conserves its energy along the trajectory, thus, its initial condition can be retrieved from any point of the trajectory. The real (dissipative) pendulum loses energy over time, and converges to 0 for $t \to \infty$, preventing inversion. 141
7.5 Gaussianization of a 1D bimodal distribution $q_0 \sim p^\star$ augmented with a Gaussian dimension $p_0 \sim \mathcal{N}(0, 1)$. Points are colored according to their probability in the final Gaussian $(q_T, p_T) \sim p_G$ distribution, as probabilities are conserved in the (p,q) space by the Hamiltonian Normalizing Flow. 143
7.6 Density estimation carried out the estimator using the HNF. As discussed in sec 7.3.1, the estimator doesn't yield the same value depdending on the p_0 value sampled. 147
7.7 *Left column:* samples from the data distribution. *Center columns:* 2D toy densities averaged over different numbers of p_0 samples . *Right column:* Kernel density estimation. 148

7.8 Generation of black & white faces at temperature of 0.7. The noise on the left is continuously transformed during to obtain the final samples. 149
7.9 *Top*: Samples from the MNIST digits dataset. *Bottom*: Random samples generated with Hamiltonian Normalizing Flow at temperature 0.7 149
7.10 Random samples generated with a convolutional HNF at temperature 0.7 150

CHAPTER 1

INTRODUCTION

At the beginning of the sixteenth century, Tommaso Masini, a collaborator of Leonardo da Vinci, stood on the roof of a building near the city of Florence. After careful study of the works of the master, he had managed to build a prototype of the first machine that would take humanity to the skies and all that remained to do was to demonstrate the capabilities of the machine. Taking a leap from the roof, he quickly realized that the machine was in fact unable to sustain his weight and fell to the ground, breaking his leg in the process. While commendable from a scientific standpoint, as this experiment could only lead to the advancement of science, he might have judged the price too high. Issues such as this one are still common to this day where, instead of a colleague's physical integrity, countless man-hours and large amounts of money must be expended to advance certain areas of science. One can think of installations such as the Large Hadron Collider and the ITER nuclear fusion project, which stand as large-scale international undertakings, involving significant budget and personnel investments. Similarly, prototyping is employed sparingly in smaller-scale applications such as the development of aviation and energetic technologies, because of very high costs.

Because of these limitations, scientists are now carrying out some of these more expensive experiments *in-silico*, using a number of numerical simulation methods. Indeed, the majority of physical phenomenons can be represented through mathematical models taking the form of a Partial Differential Equation (PDE). These models rarely have known closed-form solutions, making them hard to use directly. Fortunately, they can be solved numerically to approximate the behavior of a physical system without actually constructing it *in-vivo*.

Despite the significant advantages of this dematerialized approach to experimentation, numerical simulation is too limited for some of today's applications. Indeed, most numerical simulation methods, such as the Finite Elements

Method (*FEM*,[83]) rely on the resolution of high dimensional systems of equations, requiring significant computing power, which in turn leads to significant financial expenditure. A large body of work has been dedicated to addressing these cost issues, ranging from pure algorithmic optimization to simplified models.

As part of these efforts towards the optimization of numerical simulation methods, approaches have been developed to *reduce* physical systems to a small number of driving phenomena. With the goal of limiting the number of degrees of freedom of the systems considered, and thus the computational costs associated with their simulation. These methods are now considered to form a subfield of numerical simulation, called Reduced Order Modeling. We will show in this thesis that despite their computational advantages, the accuracy of reduced order modeling methods is often limited: considering only the dominant features of a system can lead to large approximation errors.

The aim of this doctoral thesis is to design methods for the amelioration of Reduced Order Modeling methods, improving their accuracy while retaining their low computational cost. Our ambition is to preserve the essential characteristics of applicability of the proposed methods such as interpretability, stability, or generality. To this end, we propose different data-driven approaches to improve the accuracy of the reduced equations describing the underlying physics, ranging from approximating PDEs in reduced spaces to purely data-driven methods agnostic to the underlying physics.

We note that numerical simulation methods have been intricately linked with the field of machine learning since their inception. One can think of Bayesian methods being routinely used in combination with numerical simulation tools for design optimization or the application of Kalman filters to the control of physical processes. The closure models used to simulate the Reynolds Averaged Navier Stokes equations that use data-driven methods to fit the model parameters are another example.

More recently, with the rise of neural networks, this hybridization between numerical simulation and Machine Learning has become a major topic of study. These novel models have been shown to be very flexible and able to extract complex correlations from data. Numerous applications to numerical simulation have been proposed in recent years, which we discuss at the beginning of this thesis.

In this work, we propose leveraging the approximation capabilities of neural networks to improve reduced-order modeling methods. We chose to focus on the interpretability of the resulting models, which is critical in applications where guarantees on the model are required. Because neural networks are not easily interpretable, we developed various approaches to improve this specific aspect.

To achieve our goal of interpretability, we leverage the theory of dynami-

cal systems to build theoretically sound, hybrid models. Our first proposal retains the established system equations and combines them with data-driven closures, much in the fashion of classical physical modeling methods such as the Reynolds Averaged Navier-Stokes equations. We then propose a similar method able to extract fully data-driven models from data using dynamical systems theory to derive the form of the model and improve interpretability.

Main Contributions

The main contributions of this thesis lie in the proposal of two novel methods for the reduced-order modeling of physical systems :

- **The CD-ROM method:** The method focuses on the closure of the POD-Galerkin [37, 32] reduced order modeling method. We explore the loss of information that is inherent to linear dimensionality reduction, and how it can be accounted for. We show that the method can be seamlessly used in combination with standard ODE solvers, and adapted to challenging dynamics. We demonstrate the method on various use cases used to benchmark dynamical modeling problems, such as laminar fluid flows and chaotic systems.

- **The iLED method:** This second modeling approach leverages the nonlinear dimensionality reduction capabilities of neural networks to build interpretable dynamical models. Although much more data-driven than the CD-ROM method, we show that iLED is grounded in dynamical systems theory, and highly interpretable compared to classical neural network architectures. As with the CD-ROM method, iLED is shown to perform well on various numerical simulation problems.

Outline

This thesis is organized around the various scientific communications that have been produced over the course of the Ph.D. The second chapter provides an introduction to the topic of reduced order modeling, where we introduce the most important complexities that come with the simulation of physical systems, while the third chapter emphasizes the deep connections that exist with the field of Machine Learning. Each following chapter is then based on a specific communication. We provide below a summary of each chapter:

- **Chapter 2** introduces the problem of model order reduction. A brief summary of classical numerical simulation is provided, followed by a review of the dominant dimensionality reduction approaches. We then discuss the advantages of these methods and the challenges they raise.

- **Chapter 3** introduces neural networks, powerful function approximators, which were used extensively in this thesis. Emphasis is put on the potential of neural networks for dynamical modeling and the existing applications in the literature that helped drive and improve our proposal.

- **Chapter 4** is based on the following publication :

  E. Menier, M. A. Bucci, M. Yagoubi, L. Mathelin, and M. Schoenauer, "CD-ROM: complemented deep - reduced order model", Computer Methods in Applied Mechanics and Engineering 410, 115985 (2023). https://arxiv.org/abs/2202.10746.

  This publication describes a major contribution of the thesis, the *CD-ROM* method, which is the method we developed to complement the POD-Galerkin reducer order modeling method using neural networks. This method proposes to learn a closure term for an imperfect reduced order model that results from the projection of the governing Partial Differential Equations. The final model is both accurate, based on the projected physical equations, and continuous in time.

- **Chapter 5** presents additional results obtained with the CD-ROM method. The first part of the chapter is based on the following short paper :

  E. Menier, M. A. Bucci, M. Yagoubi, L. Mathelin, T. Dairay, R. Meunier, M. Schoenauer (2022). "Continuous Methods: Adaptively intrusive reduced order model closure", Workshop on continuous time methods for machine learning ICML 2022, https://arxiv.org/abs/2211.16999.

  This contribution is an extension of the CD-ROM method, where we show that the method could be used to model strongly nonlinear dynamics. These problems are generally not straightforward to model using the POD Galerkin method as their linear projection doesn't result in a simple, low-dimensional algebraic expression. With this work, we show that the CD-ROM method can be used to model these terms while retaining the *reducible* part of the equations.

  A study on the ability of the CD-ROM model to learn the driving frequencies of a system is then presented. Further underlining the interest of the proposed architecture.

- **Chapter 6** is based on the following paper :

  E. Menier, S. Kaltenbach, M. Yagoubi, M. Schoenauer, P. Koumoutsakos (2023). "interpretable Learning of Effective Dynamics for multiscale systems", *Submitted*, https://arxiv.org/abs/2309.05812.

In this paper, we propose a method that leverages the dimensionality reduction capabilities of neural networks. The interest of nonlinear dimensionality reduction for dynamical systems is well established and also discussed in Chapter 2. Starting from this observation, we discuss the way a reduced-order modeling strategy can be constructed by extracting both a reduced manifold and a dynamical model from data using neural networks. Emphasis is put on the derivation of a theoretically grounded dynamical model in the reduced space of a neural autoencoder: In order to do so, we start from the well-studied Koopman operator [169] and leverage its links with the theory of partially observed systems (Mori-Zwanzig formalism[6]) to obtain an *ansatz* for the dynamics of the reduced system. We show that our method can transform high dimensional, non-linear PDEs into simple *quasi*-linear ODEs in suitable cases, as well as provide an interpretable framework for the study of chaotic systems.

- **Chapter 7** presents additional studies that are outside the scope of the thesis, linked to the potentiality of using dynamical systems theory to inform the construction of Deep Learning architectures. The work presented is mainly focused on generative modeling, and was presented in part in the following workshop short paper:

  E. Menier, M. A. Bucci, M. Yagoubi, L. Mathelin, M. Schoenauer (2022). "Continuous Methods: Hamiltonian Domain Translation", Workshop on continuous time methods for machine learning ICML 2022, https://arxiv.org/abs/2207.03843.

- **Chapter 8:** Conclusion of the thesis.

CHAPTER 2

REDUCED ORDER MODELING

Contents

2.1 Full order simulation 7
2.1.1 First principles 8
2.1.2 Problem Discretisation 9
2.1.3 Accuracy & Computational Cost 11
2.2 Dimensionality Reduction 13
2.2.1 Intrinsic Dimension 13
2.2.2 Linear reduction 14
2.2.3 Non Linear reduction 19
2.3 Conclusion 23

This Chapter focuses on the numerical simulation aspects of this thesis, while the next Chapter will provide the background related to neural networks and their power as flexible function approximators.

In particular, this Chapter establishes the challenges facing current model order reduction methods and the previous proposals that have been made to use data-driven approaches to improve their approximation accuracy.

2.1 Full order simulation

Full order simulation designates the range of methods based on the high-dimensional discretization of a Partial Differential Equation (*PDE*). This Section briefly introduces the methods used to solve these problems in most scientific applications, as well as the drawbacks of this approach that limit its applicability for certain critical problems.

2.1.1 First principles

The vast majority of complex physical problems can be represented by models built from first principles. That is to say, models that have been derived from physical and theoretical considerations and are expressed in the form of Partial Differential Equations as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{u}(x,t)}{\partial t} &= \mathbf{G}(\boldsymbol{u}(x,t),t),\ x \in \Omega,\ t \in [0,T], \\ \boldsymbol{u}(x,0) &= \boldsymbol{u}_0(x), \\ \boldsymbol{u}(x,t) &= f_D(x,t),\ \forall x \in \partial\Omega_D, \\ \frac{\partial \boldsymbol{u}(x,t)}{\partial x} &= f_N(x,t),\ \forall x \in \partial\Omega_N. \end{aligned} \tag{2.1}$$

Where $\boldsymbol{u}(x,t)$ is the state of the system, Ω the computational domain, $\partial\Omega_D$ and $\partial\Omega_N$ are parts of the domain's boundary $\partial\Omega$ on which the boundary conditions $f_D(x)$ and $f_N(x)$ are defined and $\boldsymbol{u}_0$ is the initial condition. Depending on the form of the operator $\mathbf{G}$, this general formulation can be used to represent the behavior of various physical problems such as quantum mechanics problems or structure simulation problems. To propose a more concrete introduction, we will focus in this Chapter on the incompressible Navier-Stokes equations, which are one model of the behavior of fluid flows at low-speed regimes.

These equations are representative of a wide variety of dynamical phenomenons encountered in more complex cases, while remaining tractable in an experimental setting. For this reason, we used them extensively as a test bed for this thesis. They are formulated as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} &= -\nabla p + \frac{1}{\mathit{Re}}\Delta \boldsymbol{u} - (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}, \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{2.2}$$

where $\boldsymbol{u}$ corresponds to the velocity field, p is the pressure field and *Re* is the Reynolds number, which is a parameter that corresponds to the degree of energy dissipation in the system. This parameter directly controls the complexity of the dynamics, which increases with its value. These equations can be used to simulate the behavior of most simple flow cases, such as the case of the flow around a cylinder, which is widely used as a benchmark for reduced-order modeling approaches. To provide a concrete example of the type of problems we focused on, a schematic of this specific case is displayed in figure 2.1.

The simulation problem consists in computing (an approximation of) the solution of Eq.(2.2).

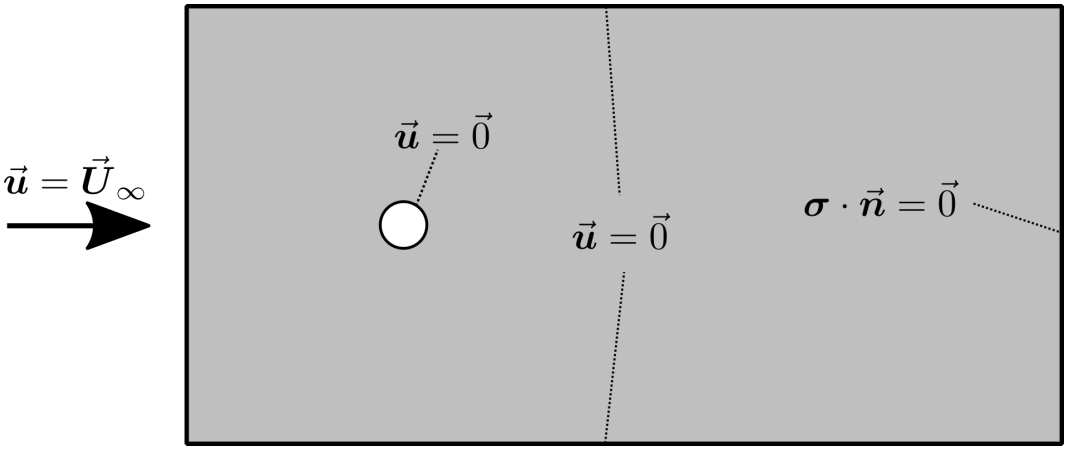


Figure 2.1: Computational domain of the cylinder flow. Typical boundary conditions for this problem are also displayed. $\vec{U}_\infty$ is the inflow velocity and σ the fluid stress tensor.

2.1.2 Problem Discretisation

The simulation problem can be solved using various discretization methods. We illustrate the procedure here with the Finite Elements Method (FEM), historically the most popular approach, and the method that was used primarily in our work. Note that other approaches are possible (Finite Differences, Spectral Elements, Finite Volumes ...), which might be more efficient depending on the simulation problem considered. However, all these methods generally face the same issues that are tackled in this work, and our results are compatible with most choices of full order discretisation method.

The Finite Elements Method proposes to divide a PDE problem such as the one in Eq.(2.1) into a number of smaller sub-problems. This is achieved by discretizing the computational domain Ω in a number of sub-domains, designated as *elements*, creating a partition of the domain, aka a *mesh*. After constructing the mesh, an approximation space for the solution of the PDE is chosen by constructing a set of basis functions ϕ_i. The solution $\boldsymbol{u}(x,t)$ is then expressed as follows:

$$\boldsymbol{u}(x,t) = \sum_{i=1}^{N} c_i(t)\phi_i(x). \tag{2.3}$$

There are a lot of choices possible for the form of the basis functions ϕ_i, they are commonly defined as piece-wise polynomial functions constructed to be non-zero on only a fraction of the mesh, typically a small number of elements. Figure 2.2 presents this idea of discretization, with a simple 1-dimensional domain divided into five sub-domains, over which the basis functions ϕ_i are defined as piecewise linear.

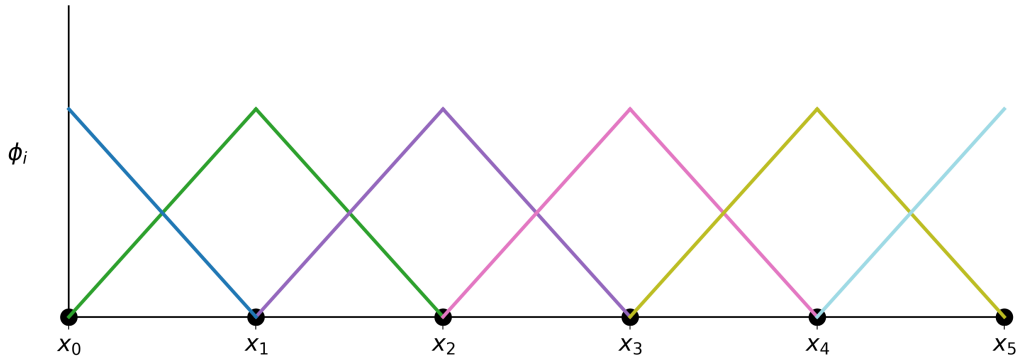


Figure 2.2: A 1-dimensional domain meshed into five sub-domains, over which six piecewise linear basis functions are defined.

The main advantage of this discretization approach is that it simplifies the computation of the spatial derivatives of the solution $\boldsymbol{u}$. Indeed, leveraging Eq.(2.3) as well as the linear nature of the derivative, we get:

$$\frac{\partial \boldsymbol{u}(x,t)}{\partial x} = \sum_{i=1}^{N} c_i(t) \frac{\partial \phi_i(x)}{\partial x}. \tag{2.4}$$

Hence the computation of spatial derivatives now entirely depends on the choice of approximation space, defined by the functions ϕ_i. This has the effect of removing spatial derivatives from a continuous PDE problem. Similarly, partial temporal derivatives simplify as follows:

$$\frac{\partial \boldsymbol{u}(x,t)}{\partial t} = \sum_{i=1}^{N} \frac{\mathrm{d}c_i(t)}{\mathrm{d}t} \phi_i(x). \tag{2.5}$$

These simplifications allow for the discretization of a continuous PDE problem such as the incompressible Navier-Stokes equations (2.2) into a system of Ordinary Differential Equations (ODEs) that determines the value of the coefficients c_i. However, the direct discretization of the PDE in Eq.(2.1) can be too constraining, depending on the form of the operator $\mathbf{G}$. The computation of second-order derivatives such as dissipative terms imposes the use of high-order polynomial basis functions ϕ_i to ensure sufficient smoothness, which in turn leads to increased computational costs. Similarly, accounting for boundary conditions is not straightforward with this direct approach. Instead, a discretization of the *weak* (or *variational* [83]) formulation of a PDE problem is often solved, to alleviate the aforementioned issues. This approach allows for the computation of the best solution in the chosen approximation space, yielding a discretized system of ODEs :

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{NS}(\boldsymbol{u}) \rightarrow \frac{\mathrm{d}c_i}{\mathrm{d}t} = \widetilde{\mathcal{NS}}(c_i), \tag{2.6}$$

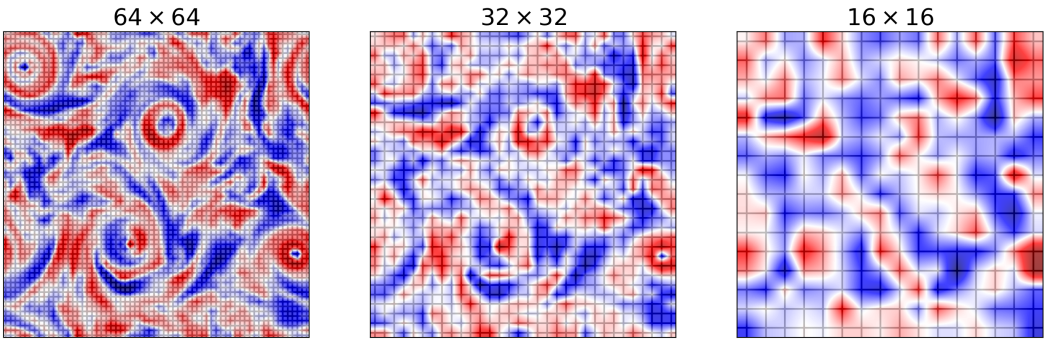


Figure 2.3: Vorticity field of the case of isotropic turbulence represented on meshes of various precision. Coarse meshes are unable to represent small-scale phenomena and filter out critical information. *Turbulence data obtained from Biferale et al. [127].*

where $\mathcal{NS}$ represents the right-hand side of the Navier Stokes equations (Eq. (2.2)) and $\widetilde{\mathcal{NS}}$ is the discretized system of N equations computed from the variational form of the Navier Stokes Equations.

Note that the goal of this Section is not to give a complete introduction to the Finite Elements method, but only to introduce some of the challenges faced by full order simulation methods. An extensive body of work has been dedicated to the Finite Elements method. We refer the reader to introductory books such as Langtangen and Mardal [83] for a detailed formal description of the method.

2.1.3 Accuracy & Computational Cost

As described in previous Section, *PDE* problems can be expressed as a system of ordinary differential equations using numerical approaches such as the Finite Elements Method. It is important to note that the resolution of the discretization (size of the largest element) is critical to the accuracy of the approximate solution. Indeed, a coarse discretization will be unable to represent complex phenomena, as illustrated in figure 2.3 with a 2D snapshot of a turbulent flow at different mesh resolutions. The figure illustrates the fact that, as the mesh grows coarser, details of the solution are filtered out. This is a significant issue as the inability of the mesh to represent the solution significantly degrades the accuracy of the computation, and might also lead to a diverging simulation. Because first principles models such as the Navier-Stokes equations often represent the balance of various quantities (*e.g.* mass or energy) in the system, their accurate representation is critical to the accurate computation of the dynamics.

A more formal expression of this idea is the Lax equivalence theorem [3], which states that for a well-posed linear PDE problem, convergence and stability of the numerical schemes are equivalent. Here, convergence refers to the

fact that, as the time-step size goes to 0, the time-discretized solution converges to the true solution. Meanwhile, stability refers to the eigenvalues of the linear operator resulting from the discretization of the PDE, stating that successful applications of the operator to advance the initial condition in time will lead to bounded solutions. In the case of PDE problems, the stability condition relates the precision of the spatial discretization (Δ_x) with the temporal discretization precision (Δ_T), ensuring that Δ_x goes to 0 with Δ_T. Equivalency between these conditions, as stated by the Lax theorem, implies that the high precision discretization of the problem is required to compute a highly accurate solution.

This means that while full order simulation methods allow for the simulation of exact models derived from first principles, **their accuracy is bounded to the precision of the discretization used to solve the problem.** Unfortunately, this parameter also controls the computational cost of solving the problem, as the number of elements, and thus the number of equations to be solved in the discretized system scales with the size of the mesh.

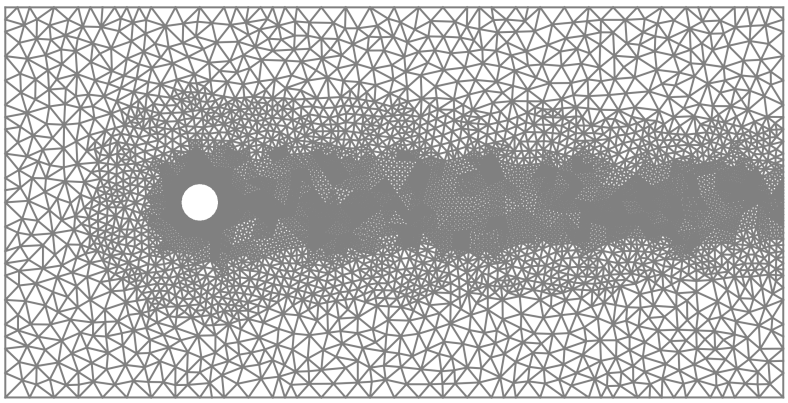


Figure 2.4: Mesh used for the simulation of the cylinder flow case.

This constraint imposes the use of high-dimensional meshes even for simple problems such as the aforementioned case of the cylinder flow. The mesh that was used in some of our work is displayed in figure 2.4, it contains approximately 12000 elements. While this is already a high-dimensional problem, it is possible to solve it in a reasonable time on a normal computer. However, real-life cases such as turbulent flows over a plane wing involve phenomena on scales separated by multiple orders of magnitude. Thus, these problems can hardly be simulated using full order simulation methods as they require meshes involving billions of elements, making them excessively expensive to solve, even on the largest computers available.

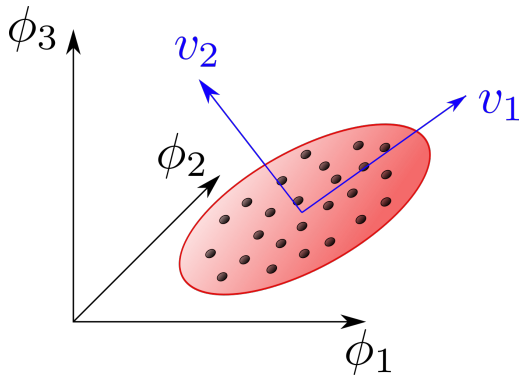


Figure 2.5: The idea of dimensionality reduction. The solution points lie on a lower dimensional (in *red*) manifold than the full space used to represent them.

2.2 Dimensionality Reduction

In the previous Section, we explained that while full order simulation methods are very accurate, due to their ability to exploit exact models, they are limited by the dimension of the discretization used to solve the problem. In this Section, we discuss the fact that the dimension of most simulation problems can be drastically reduced through various dimensionality reduction methods. A description of the dominant methods is given in the following Paragraphs, accompanied by a discussion of their advantages and drawbacks.

2.2.1 Intrinsic Dimension

Representing the state of a physical system on a very large mesh is necessary to ensure the accuracy and convergence of full order simulations. However, this high-dimensional representation is used in direct opposition to the fact that the states of most dynamical systems effectively evolve on very low-dimensional manifolds.

This idea is represented in figure 2.5, which displays a high dimensional space, defined by the directions ϕ_i. This high-dimensional space contains a lower-dimensional manifold, in red, which holds all the solution points of the problem. This is explained by the fact that a reduced number of dominant phenomena generally drives dynamical systems. These dominant phenomena can be viewed as supporting a low-dimensional manifold that optimally represents the systems. Note that the reduced representation space on figure 2.5 is presented as a linear sub-space for clarity. However, these reduced representation spaces generally correspond to non-linear *manifolds*, which do not directly correspond to low dimensional Euclidean spaces.

In fact, this idea is not limited to dynamical systems but can be framed in

the context of data compression. We often find that data correlations can be exploited to represent a dataset of N variables using a reduced number r of features, with $r << N$. This is directly due to the lack of an *optimal basis* to represent the data. An immediate parallel can for example, be drawn with the inadequacy of the pixel space to represent most image datasets.

This aspect is especially important when computational costs become critical, as it can drastically reduce the number of equations to be solved. Indeed, such low intrinsic dimensions of a system means that its evolution can be restricted to the aforementioned low dimensional, or *reduced*, space. Unfortunately, these low dimensional representation spaces are not known *a-priori*, and have to be extracted from data. To this end, a wide range of methods have been proposed. In the following Paragraphs, we provide an overview of the dominant approaches in the literature, distinguishing between two families of dimensionality reduction methods, linear and nonlinear approaches.

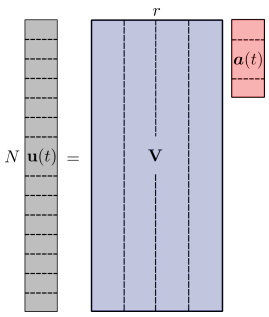
2.2.2 Linear reduction

Linear methods have been extensively studied because of their ease of use and generality. We also discuss in this Chapter their interest in terms of interpretability, as they can be seamlessly used in combination with physical equations such as Eq. (2.1).

The linear methods discussed in this Section are based on the idea of *modal decomposition*, that is to say, the decomposition of the state of a system into a linear combination of a low number r of modes $\boldsymbol{v}_i \in \mathbb{R}^N$ that represent the data more efficiently than full order discretization methods. Once identified, the modes $\boldsymbol{v}_i$ span a reduced sub-space of the high dimensional discretization of the system, so that any discretized system state $\boldsymbol{u}(t)$ lying in this space can be expressed using a reduced number of features $a_i \in \mathbb{R}$:

$$\begin{aligned} \boldsymbol{u}(t) &= \sum_{i=1}^{N} c_i(t)\phi_i, \\ &= \sum_{i=1}^{r} a_i(t)\boldsymbol{v}_i. \end{aligned} \tag{2.7}$$

With this expression, the problem is reduced to the determination of a low dimensional vector of coefficients $\boldsymbol{a}(t) = [a_1(t), a_2(t), \ldots, a_r(t)]$, which achieves computational gain when $r << N$. To simplify notations, we adopt the following matrix-vector notation:



$$\boldsymbol{u}(t) = \mathbf{V}\boldsymbol{a}(t). \tag{2.8}$$

Here, $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{N\times r}$ is a rectangular matrix assembled by stacking horizontally the modes $\boldsymbol{v}_i$, which we denote as the reduced basis. The problem then becomes one of identifying a suitable reduced basis to represent the solutions of a system. As mentioned in the previous Paragraphs, there exists a number of methods to identify this basis. In the following, we give a description of the most common methods, starting with the Proper Orthogonal Decomposition method.

Proper Orthogonal Decomposition

The proper orthogonal decomposition *POD* method ([8, 9]), also known as the principal component analysis *PCA* method[1], is a well-established algorithm. It is part of most machine learning toolkits and can be used for various applications such as data visualization or, for our purposes, dimensionality reduction.

The method proposes to extract a reduced basis of principal directions or *modes* from previously acquired system data. The method starts from a number of n_t realizations of the discretized state of the system $\boldsymbol{u}(t_i)$, arranged in a matrix $\mathbf{S} = \{\boldsymbol{u}(t_i)|i = 1, \ldots, n_t\}$, called the snapshot matrix. Computing the singular value decomposition *SVD* of the matrix $\mathbf{S} \in \mathbb{R}^{N\times n_t}$ yields three different matrices:

$$\mathbf{S} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{W}^{\mathsf{T}}. \tag{2.9}$$

Where $\mathbf{V} \in \mathbb{R}^{N\times n_t}$ and $\mathbf{W} \in \mathbb{R}^{n_t\times n_t}$ respectively hold the left and right eigenvectors of the snapshot matrix. The diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n_t\times n_t}$ holds the singular values of $\mathbf{S}$, arranged such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{n_t} \geq 0$.

The columns of $\mathbf{V}$ are the time-invariant spatial modes defining a POD basis. These POD modes are orthonormal such that $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \delta_{ij}$, with δ_{ij} the Kronecker delta and $\langle \cdot, \cdot \rangle$ is here Euclidean: $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \mathbf{v}_i^{\mathsf{T}}\, \mathbf{v}_j$. These modes are useful for dimensionality reduction because, for any $r < n_t$, the sub-space spanned by

[1]The equivalence between PCA and POD holds up to implementation details specific to the field. PCA is used in general data mining contexts while POD was specifically proposed to treat dynamical systems. However the principles behind both methods are the same.

the basis $\mathbf{V}_r = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r\}$ **optimally** approximates the data $\mathbf{S}$ over the set of $N \times r$ matrices in the sense that it minimizes the reconstruction error E_r defined as:

$$E_r = \|\mathbf{S} - \mathbf{V}_r \mathbf{V}_r^{\mathsf{T}} \mathbf{S}\|_F, \qquad \forall\, r \in [1, \ldots, n_t]\,. \tag{2.10}$$

This error can be related to the sum of the discarded singular values: $E_r^2 = \sum_{k=r+1}^{n_t} \sigma_k^2$. This implies that the information captured by the leading r modes in the POD basis can be quantified by looking at the following ratio:

$$R(r) = \frac{\sum_{k=1}^{r} \sigma_k^2}{\sum_{k=1}^{n_t} \sigma_k^2}. \tag{2.11}$$

The POD method has several advantages over other dimensionality reduction approaches, it is available in most data science packages and takes advantage of thoroughly optimized matrix manipulation algorithms. Moreover, **the criterion derived in the equation above allows for the *a-priori* estimation of the accuracy of the reduced basis.** Choosing r so that $R(r) = 1$ yields a perfectly accurate representation of the data in the snapshot matrix S. We also note that $R(r) = 1$ for $r = \min(n_T, N)$, thus, in cases where the number of snapshots n_T is lower than the dimension of the discretization (N), as is often the case in complex simulation problems, the POD method can be used as a *lossless* reduction method. This criterion also provides a direct interpretation of the method. Indeed, the **POD modes have a clear statistical meaning** as they directly relate to the optimal representation of the available data.

Another advantage of the method is that it conserves linear invariants of the data. That is to say, any linear condition verified by the data samples $\boldsymbol{u}(t)$ is also verified by the POD modes $\boldsymbol{v}_i$:

$$\mathbf{B}\boldsymbol{u}(t_i) = 0 \implies \mathbf{B}\boldsymbol{v}_i = 0. \tag{2.12}$$

With $\mathbf{B} \in \mathbb{R}^{N \times N}$ a matrix that encodes the linear relationships verified by the data samples. This is explained by the fact that the POD modes $\boldsymbol{v}_i$ lie in the span of the columns of the matrix $\mathbf{S}$, which correspond to the data samples. Thus, they can be expressed as a linear combination of the data samples:

$$\begin{aligned} \boldsymbol{v}_i &= \sum_{j=1}^{n_t} c_{v_i,j} \boldsymbol{u}(t_j), \\ \implies \mathbf{B}\boldsymbol{v}_i &= \sum_{j=1}^{n_t} c_{v_i,j} \underbrace{\mathbf{B}\boldsymbol{u}(t_j)}_{0} = 0. \end{aligned} \tag{2.13}$$

Finally, we see that any linear combination of the POD modes $\boldsymbol{u}(t) = \mathbf{V}\boldsymbol{a}(t)$ also verifies the condition in Eq. (2.12). This means that certain physical con-

straints such as the conservation of mass for incompressible flows are encoded in the reduced basis:

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.14}$$

Because the samples $\boldsymbol{u}(t_i)$ used to compute the POD basis $\mathbf{V}$ all verify the conservation of mass, any reconstruction on the POD basis will also respect this physical constraint. Indeed, the divergence $\nabla\cdot$ is a linear operator. This particular point provides additional physical guarantees that can be hard to obtain with other reduction methods. Because of these advantages, the method has been used to study complex systems, such as fluid flows, for more than 30 years ([31, 17]).

The method does have significant drawbacks, however, most notably, optimality in the sense of the reconstruction error does not imply that the basis can efficiently capture the dynamics of the system. Concretely, small reconstruction errors often have a significant impact on the dynamics of a system and can compound over time, leading to inaccurate reduced order models (*ROMs*). This is discussed in more detail in Section 3.2, while the following Paragraphs discuss methods that might be better suited for the representation of system dynamics.

DMD

Dynamic mode decomposition or *DMD* ([42]) is a second well-established method for the identification of dominant modes from system data. Where the POD method uses the L_2 reconstruction error (Eq. (2.10)) to construct the modes. The DMD is rooted in Koopman theory, as it assumes a linear relationship between temporally ordered data:

$$\boldsymbol{u}_{t_2} = \mathbf{A}\boldsymbol{u}_{t_1}. \tag{2.15}$$

With $\mathbf{A}$ a matrix whose eigenvectors are the DMD modes $\boldsymbol{v}_j$. Note that the linear relationship in Eq. (2.15) is equivalent to expressing the time evolution of the solution $\boldsymbol{u}(t)$ as a superposition of the eigenvectors of the matrix $\mathbf{A}$ such that:

$$\boldsymbol{u}(t) = \sum_{j=1}^{N} c_j e^{\lambda_j t}\boldsymbol{v}_j. \tag{2.16}$$

With $Re(\lambda_j)$ and $Im(\lambda_j)$ the growth rate and frequency associated with the eigenvectors $\boldsymbol{v}_j$ of the matrix $\mathbf{A}$. With this result, we see that Dynamic Mode Decomposition identifies both a basis on which the state of the system of interest can be expressed and a representation of the system dynamics. This is an important aspect as it suggests that the modes identified by the DMD

method might be better suited to support the dynamics than POD modes. In other words, where the POD modes have statistical meaning in the sense that they optimally represent the data in the snapshot matrix, the **DMD modes have dynamical meaning** as they are associated with specific frequencies in the system data.

The goal of the DMD method is the identification of the matrix $\mathbf{A}$. Over the years, multiple variants of the algorithm have been proposed. We refer the reader to Tu et al. [66] for a description of these variants and a discussion of their advantages and drawbacks while we describe the general method in this Paragraph. The method starts with pairs of data points $\{(\boldsymbol{u}_k, \boldsymbol{u}_k^{\#})\}, k = 1, \dots, K$, with an assumed linear relationship (as in Eq. (2.15)). Note that the case of temporally ordered data is a particular case of this more general framework, as we can identify $\boldsymbol{u}_k = \boldsymbol{u}_{t_1}$ and $\boldsymbol{u}^{\#} = \boldsymbol{u}_{t_2}$. A major advantage of this general formulation is that it can accommodate samples extracted from different trajectories, but representative of the same system. After assembling the matrices $\mathbf{U} = \{\boldsymbol{u}_k | k = 1, \dots, K\}$ and $\mathbf{U}^{\#} = \{\boldsymbol{u}_k^{\#} | k = 1, \dots, K\}$, the following problem is solved:

$$\begin{aligned} \mathbf{U}^{\#} &= \mathbf{A}\mathbf{U}, \\ \implies \mathbf{A} &= \mathbf{U}^{\#}\mathbf{U}^{+}. \end{aligned} \tag{2.17}$$

With $\mathbf{U}^{+}$ the pseudo-inverse of the matrix $\mathbf{U}$. Thus, the matrix $\mathbf{A}$ corresponds to the solution of a least-square regression problem. By computing its eigenvectors $\boldsymbol{v}_j$, the DMD modes are obtained.

Contrary to the POD method, there is no clear metric to order and select the most relevant DMD modes. A number of criteria can be used to select a reduced number r of DMD modes $\boldsymbol{v}_j$ to reduce the dimension of the system of interest. The modes can be selected based on their norm, although some care should be given to the scaling of the modes ([66]). They can also be selected according to their associated frequencies and growth rate to focus on different aspects of the dynamics, *i.e.*, high frequencies might be filtered out to focus on larger scale phenomena in the system. Another approach proposed is to look for a sparse representation of the snapshots $\boldsymbol{u}_k$ in the basis of the DMD modes $\boldsymbol{v}_j$ to identify a reduced number of expressive modes ([62]).

Applications of linear reduction

The methods derived above have become ubiquitous for the analysis of dynamical systems and they have extended and applied to a wide range of problems. Their ability to identify particular modes associated with clear interpretations has led to them being used to gain insights into the physics of dynamical systems ([155, 139, 55]). We will show later in the thesis that they can also be used to build dynamical models and forecast the behavior of a system outside of the

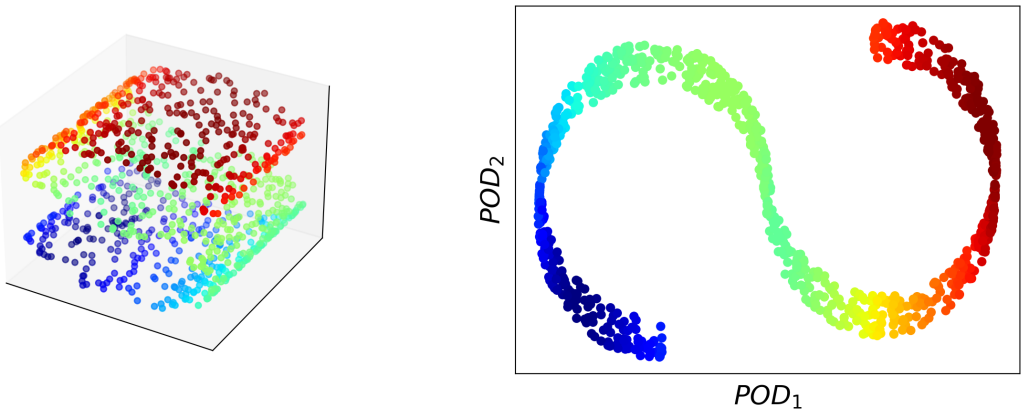


Figure 2.6: *Left:* Points lying on a two-dimensional manifold, embedded in a three-dimensional space. *Right:* Projection of the data on the leading two POD modes.

conditions used to extract the linear basis of modes. Finally, we mention various extensions that have been proposed to apply linear reduction methods to control problems. Methods such as the Balanced Proper Orthogonal Decomposition *BPOD* ([35]) were specifically designed to handle data obtained through sparse measurements of a system and account for the particularities of the control of dynamical systems. We refer the reader to more thorough reviews on the topic ([94, 68]) for more detailed discussions.

2.2.3 Non Linear reduction

Because of their simplicity and stability, linear methods are very efficient and widely used for dimensionality reduction. However, they are overtaken in terms of efficiency by non-linear reduction methods. This is due to the fact that the states of dynamical systems do not directly lie in well-organized linear subspaces as depicted in figure 2.5, but on non-linear *manifolds* which have a low intrinsic dimension, but aren't well captured by linear reduction methods.

To illustrate this phenomenon, we use the case of a simple 2D manifold in a three-dimensional space, depicted in figure 2.6 and show that linear dimensionality reduction through POD fails to capture the two-dimensional manifold as the depth of the data points is lost and they are overlaid in the same place in the low-dimensional representation. This simple example outlines the limitations of linear dimensionality reduction, and the potential for improvement using non-linear reduction methods.

Various methods have been developed over the years to re-arrange non-linear manifolds in low-dimensional sub-spaces. We introduce a few of them in

the following to provide some insights into their potential and limitations[2].

Locally Linear Embeddings

Locally Linear Embeddings are a nonlinear dimensionality reduction method that leverages the local *flatness* of the manifold to be identified. The method assumes that small regions of the manifold can be seen as local linear spaces. This means that data points $\boldsymbol{u}$ can be expressed as linear combinations of their neighbors:

$$\boldsymbol{u}_i = \sum_{j=1}^{K} w_j \boldsymbol{u}_j. \tag{2.18}$$

Where $\boldsymbol{u}_j$ are the K nearest neighbors of the point $\boldsymbol{u}_i$. The method computes a set of K weights for every data point, under the constraint that each set of weights w_j sums to 1, which is equivalent to minimizing the following objective:

$$J_{1,i} = \| \sum_{j=1}^{K} w_j (\boldsymbol{u}_j - \boldsymbol{u}_i) \|^2. \tag{2.19}$$

These weights represent the local relationships between points on the manifold. These relationships can be expected to be conserved as best as possible in any lower-dimensional representation space. Thus, the method looks for low-dimensional coordinates $Y_i = f(\boldsymbol{u}_i)$ that minimise the following cost:

$$J_{2,i} = \| \sum_{j=1}^{K} Y_i - w_j Y_j \|^2 \tag{2.20}$$

This defines a quadratic form in the low dimensional coordinates Y_i which can be minimized to obtain the coordinates of the data points in the *latent*, low dimensional space. Figure 2.7 displays the results obtained with this method on the simple two-dimensional manifold of figure 2.6.

As shown in the figure, the LLE method is able to exploit the locally linear structure of the manifold to compute *global* coordinates for each data point. The points are well organized in the two-dimensional latent space as the manifold is "unrolled".

Isomaps

Similar to locally linear embeddings, Isomaps are a non-linear dimensionality reduction method that constructs a low dimensional embedding of each point

[2]Code for the examples shown in this Section is available at https://github.com/emenier/manifolds

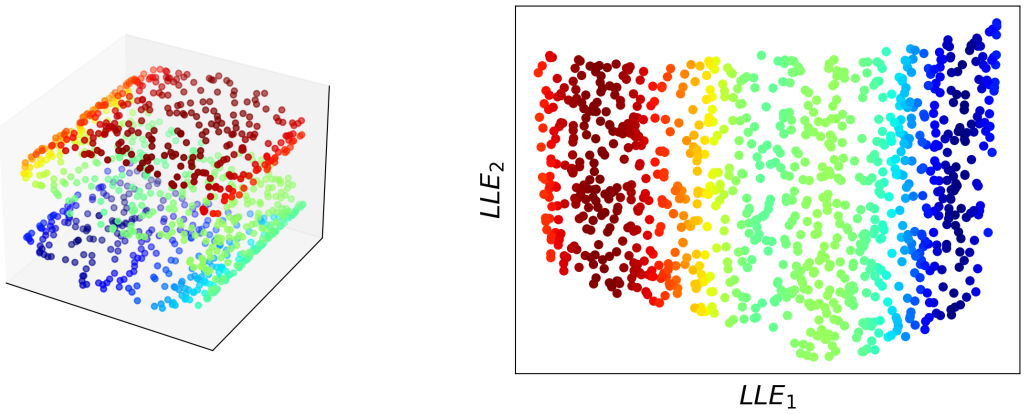


Figure 2.7: *Left:* Points lying on a two-dimensional manifold, embedded in a three- dimensional space. *Right:* Two-dimensional embedding of the data, obtained with the LLE algorithm.

while respecting certain properties of the dataset. Isomaps are a form of Multi Dimensional Scaling ([19]), while Locally Linear Embeddings leverage the local relationships between data points to learn the shape of the data manifold, Multi Dimensional Scaling *MDS* looks for an embedding that conserves the distances between each point in the dataset. That is to say, the coordinates (Y_i) derived through MDS minimize the following loss function, denoted as the Stress:

$$\text{Stress}_D(Y_1, Y_2, \ldots, Y_n) = \sqrt{\sum_{i \neq j = 1,\ldots,N} (d_{ij} - \|Y_i - Y_j\|)^2}. \tag{2.21}$$

Where d_{ij} is the distance between the data points $\boldsymbol{u}_i$. The particularity of Isomaps lies in the choice of distance metric d_{ij}. While MDS generally uses the Euclidean distance to compute the low dimensional embedding, Isomaps use the geodesic distance on the data manifold. This distance can be estimated by constructing a graph of the nearest neighbors of each data point and using a shortest path algorithm to approximate the shortest distance on the manifold. Once this distance is computed, the stress function is minimized to obtain the reduced embedding of the data. Figure 2.8 presents the results obtained with the approach.

As with locally linear embeddings, a well-organized latent representation of the data is obtained, exploiting the relationships between the data points while accounting for the shape of the manifold. It can be argued that Isomaps tend to yield a better-organized representation of the data. However, this comes at a cost as the minimization of the *stress* (Eq. (2.21)) becomes expensive to carry out when applied to large datasets.

These two examples show that non-linear dimensionality reduction can perform better than linear methods as it focuses on learning the shape of the

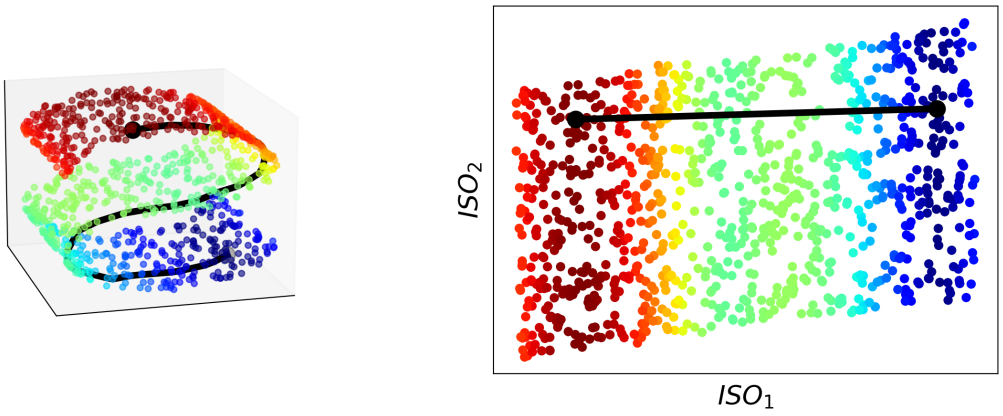


Figure 2.8: *Left:* Points lying on a two-dimensional manifold, embedded in a three- dimensional space. *Right:* Two-dimensional embedding of the data, obtained with Isomaps. The conservation of the geodesic distance is also illustrated in *black* with the shortest path between two points on the 3D manifold presented on the left, and the equivalent Euclidean distance in the reduced space on the right.

underlying manifold. Note that there exist various other methods of nonlinear dimensionality reduction, such as t-distributed stochastic neighbor embeddings (t-SNE, [41]), or diffusion maps ([34]) which have been applied in the context of dynamical systems ([102]). Kernel methods have also been used to efficiently compute and exploit non-linear transformations of a system's state ([26]). Spectral sub-manifolds (*SSM*) are another promising research track for the non-linear discovery of underlying manifolds. SSMs have also been applied to dynamical modeling problems ([159, 81]).

Limitations

The last Paragraphs showed that nonlinear reduction methods can exploit the curved nature of most data manifolds. They have been applied to dynamical systems modeling ([185, 102]) however, they have a number of limitations that should not be overlooked.

While we showed that nonlinear reduction was very efficient at capturing the low dimensional relationships between data points, it is often unclear how the original data can be reconstructed from the low-dimensional embeddings. This effectively limits the efficiency of these methods in reduced order modeling applications, as the goal is often to reconstruct the state of the system at selected time steps, or under certain conditions, which goes beyond the computation of a low-dimensional representation. Previously cited methods often rely on the development of an additional model to reconstruct the full-order state of the system from its low dimensional representation. While this can be an effective

approach, it implies additional modeling work, often combining methods that were not designed for this purpose.

2.3 Conclusion

This Chapter discussed the topic of dimensionality reduction for numerical simulation. We showed that the full order simulation methods used to numerically solve PDE problems were generally tied with excessive computational costs, partly due to their inability to account for the structure of the manifolds supporting the dynamics of the systems under study. We also discussed the various approaches that could be taken to identify these manifolds, in the hope of determining better representations, yielding additional insights in the dynamics under study and reducing the dimensionality of the problems to be solved.

Most importantly, we showed that despite their numerous advantages, linear reduction methods were generally unable to optimally represent low dimensional representation spaces in the context of dynamical modeling. This aspect motivated the introduction of non-linear reduction methods, which also come with certain limitations. In the following Chapter, we introduce neural networks, a type of machine learning algorithm that was used extensively in this work, and discuss their advantages in the context of both nonlinear dimensionality reduction as well as dynamical modeling.

CHAPTER 3

NEURAL NETWORKS FOR DYNAMICAL MODELING

Contents

3.1 Introduction to Deep Learning 26
3.1.1 The Multi Layer Perceptron 27
3.1.2 Backpropagation 27
3.1.3 Stochastic Gradient Descent 30
3.1.4 Inductive Bias 32
3.1.5 Neural networks for dimensionality reduction 33
3.2 Learning the Dynamics 34
3.2.1 Fully data-driven models 36
3.2.2 On the importance of Hybridisation 41
3.2.3 Deep Learning and physics 42
3.2.4 Summary 49
3.3 The POD Galerkin method 49
3.4 Conclusion 53

This Chapter introduces neural networks, focusing on Deep Neural Networks, which we chose to use to approximate the various operators and maps that form the basis of the methods proposed in this thesis. Indeed, neural networks have several properties that make them a suitable choice for dynamical modeling. They are universal approximators, meaning that they can approximate any continuous function provided they have a sufficient amount of trainable parameters (see Section 3.1.1), we note that increases in model expressivity do not come with exploding computational costs, as is the case with approaches

such as polynomial regression methods (see Section 3.2.1). They belong to the class of differentiable programming methods [96], meaning that their gradients can be evaluated using automatic differentiation, allowing for their use in combination with gradient-based optimization methods (see Section 3.1.2). This aspect also makes neural networks particularly suitable for the resolution of PDE problems, as has been shown with certain approaches that propose to directly learn the solutions of PDEs by minimizing the residual of the governing equations (see Section 3.2.2). The next Section first introduces the basic aspects of building and training neural networks, while Section 3.2 describes some of the approaches that have been proposed to directly extract the dynamics of physical systems from data using Neural Networks.

3.1 Introduction to Deep Learning

Part of the family of Machine Learning methods, Artificial Neural Networks have been in development for more than 60 years, when one of the first studies training a Multi Layer Perceptron (*MLP*) using stochastic gradient descent was published [7]. Unfortunately, the limited computing power available at the time proved to be an obstacle to their widespread adoption, despite the development of seminal approaches. We can cite, for example, the use of Convolutional Neural Networks for handwritten digit recognition [16], the use of neuroevolution for control [21], as well as the early application of *MLPs* to physical simulation problems [30]. It is not until very recently, with the development of heavily parallelized computation leveraging the capabilities of Graphical Processing Units (*GPU*) that neural networks became the algorithm of choice for the handling of large datasets to learn complex tasks. The seminal work of Krizhevsky, Sutskever, and Hinton [49], which overtook every other method available at the time on the famous *ImageNet* image recognition competition, is often cited as the starting point of the widespread adoption of deep neural networks (*DNNs*) in the Machine Learning community.

With the growing availability of computing power and their unmatched performance on most complex learning problems, the research interest in neural networks has grown exponentially. This has led to the development of unprecedented applications, especially in the field of Natural Language Processing with the recently released Large Language Models. These models established the ability of neural networks to leverage enormous amounts of data into creating versatile human conversation engines [194, 195, 190]. In parallel, Neural Networks are now being applied to most scientific problems with varying degrees of success, from robotics [196] to drug discovery [145].

The study of Artificial Neural Networks, and in particular Deep Neural Networks, is now designated as a standalone field, also called Deep Learning. In the following Paragraphs, we give an introduction to the building blocks of the

most simple Deep Learning model, the Multi Layer Perceptron, which we used extensively over the course of this thesis.

3.1.1 The Multi Layer Perceptron

The Multi Layer Perceptron is one of the most simple Deep Learning model available, it is constructed as a succession of trainable operations, called *layers*:

$$\Phi_\theta(x) = \phi_{N,\theta} \circ \ldots \circ \phi_{3,\theta} \circ \phi_{2,\theta} \circ \phi_{1,\theta}(x). \tag{3.1}$$

In the above expression, Φ_θ represents the complete network, x is the input of the network, $\phi_{i,\theta}$ are the layers, and θ represents the set of learnable parameters, also called *weights*. In the case of a Multi Layer Perceptron, the layers are constructed as so-called *fully-connected* layers, i.e., each $\phi_{i,\theta}$ is the composition of a nonlinear function and a trainable affine transformation of its inputs:

$$\phi_{i,\theta}(x) = \sigma(A_{i,\theta}x + B_{i,\theta}) \tag{3.2}$$

where $A_{i,\theta} \in \mathbb{R}^{d_{\phi_{i+1}} \times d_{\phi_i}}$ is a trainable weight matrix and $B_{i,\theta} \in \mathbb{R}^{d_{\phi_{i+1}}}$ a trainable weight vector designated as the bias. Finally, σ is a non-linear function, called the activation function that is defined at the construction of the neural network. Popular choices for σ are the sigmoid function, or the rectified linear unit (*ReLU*, [111, 173]), except for the last layer ϕ_N, where σ is generally defined as the identity.

It has been shown that under mild conditions on the choice of σ, a neural network using only one hidden layer (*i.e.* $\Phi(x) = \phi_2 \circ \phi_1(x)$) can approximate any continuous function, provided the width of the hidden layer d_{ϕ_1} is sufficiently large [22]. However, building arbitrarily deep networks by stacking more and more layers has also been shown to increase the expressivity of neural networks, without requiring exponentially high layer dimensions [104]. Thus, most multi layer perceptrons are constructed as a sequence of more than two reasonably *wide* layers, as shown in figure 3.1.

3.1.2 Backpropagation

The MLP introduced in the previous Section is a simple and flexible network architecture that can be trained to approximate any continuous function from data. To do so, the set of weights $\theta = \{(A_i, B_i) | i = 1, \ldots, N\}$ is iteratively modified to minimize an objective function, called the *loss function*.

This loss function can take many forms depending on the task to be learned, as long as it is differentiable. To give a simple example, in a simple regression framework, we are given data pairs (x_i, y_i) and we want to learn the map $\Phi : \mathbb{R}^{d_x} \mapsto \mathbb{R}^{d_y}$ between the inputs x and outputs y. This can be done by minimizing the mean squared error (*MSE*) between the output of the neural network $\Phi(x_i)$

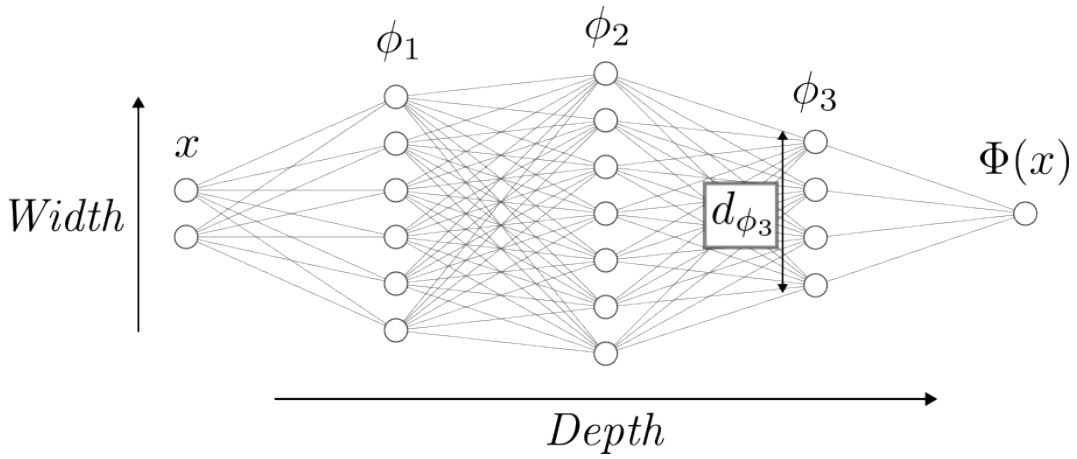


Figure 3.1: Representation of a Multi Layer Perceptron ($\Phi(x)$) with 3 hidden layers.

and the true value y_i. In this case, the loss function ($\mathcal{L}$) takes the simple following form:

$$\mathcal{L} = \frac{1}{n_x} \sum_{i=1}^{n_x} \|\Phi(x_i) - y_i\|_2^2. \tag{3.3}$$

In deep learning, the above loss is generally optimized through some variant of the basic gradient descent algorithm: the parameters θ of the neural network are iteratively adjusted in the opposite direction of the gradient of the loss $\mathcal{L}$ with respect to the parameters as follows:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial \mathcal{L}}{\partial \theta} \tag{3.4}$$

where $\eta \in \mathbb{R}$ is the so-called *learning rate* that controls the gradient step size. Note that gradient descent is one of the simplest optimization strategies available for this kind of problem and can lead to slow optimization as well as local optima. Both issues that could be addressed using more complex approaches, such as second-order optimization methods. However, the particularity of neural networks is that they rely on large amounts of trainable parameters, which makes the computation of second-order derivatives very expensive.

The evaluation of the gradient of the loss in Eq. (3.4) with respect to the parameters is made possible thanks to the particular form of the network, using iteratively the chain rule for differentiation: this leads to the algorithm called *backpropagation*, and can be easily programmed using reverse mode auto-differentiation. Introduced in Rumelhart, Hinton, and Williams [15] in the context of neural networks, the backpropagation algorithm can be seen as an adaptation of an adjoint optimization algorithm proposed earlier in Pontriagin et al. [4].

We give below a general derivation of the algorithm. It allows for the efficient computation of the gradient of the loss in Eq. (3.3) with respect to the set of weights θ of the network:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} = \frac{\partial \mathcal{L}}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial z_N}\frac{\partial z_N}{\partial \theta}. \tag{3.5}$$

The notation $z_i,\ i = 0, \dots N$ is introduced to denote the intermediate state of the network. That is the output of each layer, with $z_0 = x$ and $z_N = \Phi(x)$. To compute the gradient $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta}$, the above equation has to be simplified as the term $\frac{\partial z_N}{\partial \theta}$ can be expensive to estimate directly. Indeed, it captures the impact of any parameter change in the earlier layers on the output of the last layer. The idea of the backpropagation algorithm is to avoid computing this term through the introduction of a set of multipliers $\mu_i,\ i = 1, \dots, N$:

$$\mathcal{J} = \mathcal{L} + \sum_{i=1}^{N} \mu_i^\mathsf{T} (\underbrace{z_i - \phi_i(z_{i-1})}_{0}), \tag{3.6}$$

$$\implies \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta}. \tag{3.7}$$

Through manipulations of Eq. (3.6), conditions on the values of the multipliers μ_i can be derived to avoid computing the intermediate gradients $\frac{\partial z_i}{\partial \theta}$:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta} =& \frac{\partial \mathcal{J}}{\partial z_N}\frac{\partial z_N}{\partial \theta} + \sum_{1}^{N} \mu_i^\mathsf{T} \frac{\partial z_i}{\partial \theta} \\ &- \sum_{1}^{N} \mu_i^\mathsf{T} \frac{\partial \phi_i(z_{i-1})}{\partial z_{i-1}}\frac{\partial z_{i-1}}{\partial \theta} \\ &- \sum_{1}^{N} \mu_i^\mathsf{T} \frac{\partial \phi_i(z_{i-1})}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial \theta}.\end{aligned} \tag{3.8}$$

Taking out the N_{th} term of the first sum, and observing that $\frac{\partial z_0}{\partial \theta} = 0$, yields the following:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta} =& \left(\frac{\partial \mathcal{J}}{\partial z_N} + \mu_N^\mathsf{T}\right)\frac{\partial z_N}{\partial \theta} \\ &+ \sum_{1}^{N-1} \left(\mu_i^\mathsf{T} - \mu_{i+1}^\mathsf{T}\frac{\partial \phi_{i+1}(z_i)}{\partial z_i}\right)\frac{\partial z_i}{\partial \theta} \\ &- \sum_{1}^{N} \mu_i^\mathsf{T} \frac{\partial \phi_i(z_{i-1})}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial \theta}.\end{aligned} \tag{3.9}$$

With these manipulations, the conditions on the values of μ_i are clearly identified:

$$\mu_N = -\frac{\partial \mathcal{J}}{\partial z_N}^\mathsf{T}, \tag{3.10}$$

$$\mu_i = \frac{\partial \phi_{i+1}(z_i)}{\partial z_i}^\mathsf{T} \mu_{i+1}. \tag{3.11}$$

The advantage of these conditions in the context of neural networks is that the jacobians $\frac{\partial \phi_{i+1}}{\partial z_i}$ are easily computed as the layers ϕ_i are generally based on simple expressions. In the case of Multi Layer Perceptrons, they simply correspond to a matrix multiplication followed by a non-linear function. Computing the multipliers μ_i such that they verify the above conditions effectively cancels out the intermediate states' derivatives $\frac{\partial z_i}{\partial \theta}$ in the expression of the gradient (Eq. (3.9)). Finally, the gradient of the loss with respect to the network parameters is estimated from the values of the multipliers:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} = -\sum_1^N \mu_i^\mathsf{T} \frac{\partial \phi_i(z_{i-1})}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial \theta}. \tag{3.12}$$

Once again, the structure of the network can be exploited to simplify the above computation, as each layer ϕ_i is only dependent on a small subset of θ, meaning that the jacobian matrices $\frac{\partial \phi_i}{\partial \theta}$ are very sparse and don't need to be fully assembled. For illustration purposes, we provide a minimal implementation example[1].

This algorithm can be extended to compute the gradient of much more complex network architectures. Significant effort has been devoted to the development of optimized and flexible frameworks that implement reverse-mode automatic differentiation, we cite for example TensorFlow [71], JAX [98], and Pytorch [122] which is the framework that was used extensively in this thesis.

3.1.3 Stochastic Gradient Descent

Neural Networks are able to handle large datasets during training because they work extremely well in combination with Stochastic Gradient Descent (*SGD*). The basic idea of SGD is to only evaluate the gradient of the model on a random subset of the data at every step rather than the full dataset. This has the advantage of reducing the cost of each gradient step, as the above equations are only computed for a reduced number of data samples. More importanty, this reduces the memory footprint of the algorithm. This approach introduces a degree of noisiness in the estimation of the gradient, which has been shown to in fact improve the efficiency of the gradient descent method.

[1] https://github.com/emenier/backpropagation

Schemes have been developed to leverage the noisiness of the gradient to further improve the optimization of neural networks. Most of these schemes are based on *momentum*, which in the simplest case corresponds to computing a running average of the gradient and using this average to compute the weight update:

$$m_t = \beta m_{t-1} + (1-\beta)\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_{t-1}}, \tag{3.13}$$

$$\theta_t = \theta_{t-1} - \eta\, m_t. \tag{3.14}$$

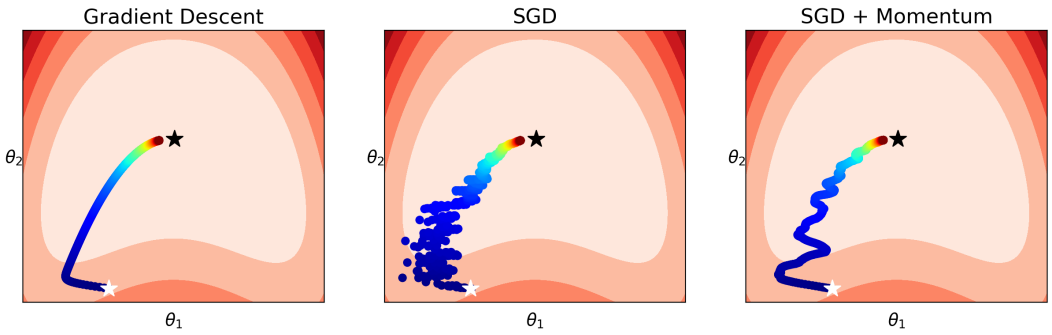
With $\eta \in \mathbb{R}$ the learning rate and $\beta \in [0,1]$ the rate of change of the momentum m_t. This type of scheme is now considered state of the art for the optimization of neural networks, as it has been shown to find better optima, with better generalization properties, in the parameter spaces of neural networks. One interpretation of the scheme in Eq.(3.13) is to see it as a filtering of the gradients' trajectory. That is to say, high-frequency variations from one gradient step t to another $(t+1)$ are attenuated in the trajectory of m_t as they cancel each other out. Meanwhile, dominating directions in the gradient from one random batch of data samples to another are reinforced and lead to faster descent in these directions.

Figure 3.2 presents the optimization path followed by the variants of the Gradient Descent algorithm on a simple two-parameter case. The figure shows that computing the gradient from the full dataset is wasteful as a stochastic estimation from two samples at each step yields similar performance. The interest of using momentum in combination with SGD is also displayed, as some of the noisiness in the gradient is filtered out, leading to a much smoother optimization path[2].

Note that this momentum scheme is only presented as an illustration, as better refined variations of this concept are used in practice, we cite for example Nesterov momentum [78] that is often used in combination with SGD. Other schemes such as RMSProp [52] compute the second order momentum of the gradient to scale the gradient step at each iteration. The Adam optimizer [87] which was used extensively in this work uses both the first and second order momentum to compute the gradient update :

$$\begin{aligned} m_t &= \beta m_{t-1} + (1-\beta_1)\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_{t-1}}, \\ s_t &= \beta m_{t-1} + (1-\beta_2)\left(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_{t-1}}\right)^2, \\ \theta_t &= \theta_{t-1} - \eta\frac{m_t}{\sqrt{s_t}}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_{t-1}}. \end{aligned} \tag{3.15}$$

[2]SGD code example available at https://github.com/emenier/SGD



	Number of function evaluations
Gradient Descent	1.5×10^7
SGD (and momentum)	3×10^3

Figure 3.2: Example of various Gradient Descent strategies to fit the parameters of the Rosenbrock function ($f(x, y) = (\theta_1 - x)^2 + \theta_2(y - x^2)^2$). *Left:* Gradient descent on the full dataset. *Middle:* Stochastic Gradient Descent using mini-batches of two samples. *Right:* Stochastic Gradient Descent with momentum using the same mini-batch size. The learning rate η and the number of gradient steps taken are the same. Level sets of the loss function are shown in red.

Where s_t is the second order momentum of the gradient, and $\beta_1, \beta_2 \in \mathbb{R}$ are user-defined parameters that control the rate of change of the first and second order momentum. With this formulation, the gradient is scaled at each iteration, where the simpler momentum scheme of Eq.(3.13) proposed to advance the parameter in the direction of the first moment of the gradient. The idea behind both methods remains similar, as they both leverage previous values of the gradient to alleviate the impact of the noisiness of the gradient inherent to SGD.

3.1.4 Inductive Bias

Before concluding this short introduction to Deep Learning, we discuss the topic of inductive bias. The multi layer perceptron architecture which we presented in the above Paragraphs is the simplest existing Neural Network architecture. However, the basic concepts that are used to build and train this architecture have been improved upon and extended to propose different architectures, better suited to specific learning tasks. Indeed, the *fully connected layer* which is the basic building block of the MLP learns linear combinations of all the dimensions of the previous layers' state. This is a very flexible approach as it doesn't assume any specific relationships between features of the input. However, it can lead to harder optimization, waste of compute, and doesn't always yield the desired behavior.

A simple example is the case of image data where spatially close features

(pixels in this case) are strongly correlated and can be analyzed locally to extract information. Using fully connected layers on this type of data implies that the full image is considered for every dimension of a layer's output, which leads to an exponentially high number of parameters, and considers interactions between far apart pixels which does not generally provide useful information. Even worse, this type of approach is very sensitive to translations of the input, meaning that shifting an image by a few pixels can drastically change the output of the neural network if it is not carefully trained.

To address this issue, *inductive biases* can be embedded in the network architecture. In the case of Image data, convolutional layers were developed to efficiently extract local information from images [16]. These layers learn relatively small convolutional filters that learn to exploit local correlations in images, leading to translation equivariant layers and much more efficient computations. Similarly, *Recurrent Neural Networks* (*RNN*) [23, 56] were developed to treat time series data through memory mechanisms. RNNs are discussed in more detail in Section 3.2.1.

This idea of inductive bias has also been applied to the case of physical simulations, where Graph Neural Networks, an architecture specially designed to handle graph data, was applied to simulation problems [166]. Networks with equivariance properties with respect to certain groups, such as the group of three-dimensional rotations, have also been developed for the analysis and simulation of molecular dynamics [168].

3.1.5 Neural networks for dimensionality reduction

Neural Networks can be used to learn continuous functions from data, which has led to their natural application to dimensionality reduction tasks. Indeed, the various methods presented in Section 2.2 share the same goal of learning a map $\mathbf{\Phi}$ from data, such that $z = \mathbf{\Phi}(\mathbf{u})$ with $d_z \ll d_x$. Which is generally done by minimizing a suitable criterion to ensure the learned embedding respects the properties of the original manifold.

For most of the nonlinear reduction methods introduced (Isomaps, Locally Linear Embeddings..), the aforementioned criterion is based on the conservation of the local relationships between data points in the low dimensional embedding. In the case of neural networks, the model is trained to reconstruct the data points, passed through a *bottleneck* layer ϕ_z of dimension d_z. Thus, the network learns to optimize the intermediate representation $\phi_z(\mathbf{u})$ so that it contains sufficiently enough information to reconstruct the original data points $\boldsymbol{u}$, or a good approximation thereof. This architecture is called an Autoencoder and is built with two different networks, the Encoder $\mathcal{E}$ which learns the map $z = \mathcal{E}(\mathbf{u})$, and the decoder $\mathcal{D}$ which learns the reconstruction of the data $\boldsymbol{u} \approx \mathcal{D}(z)$.

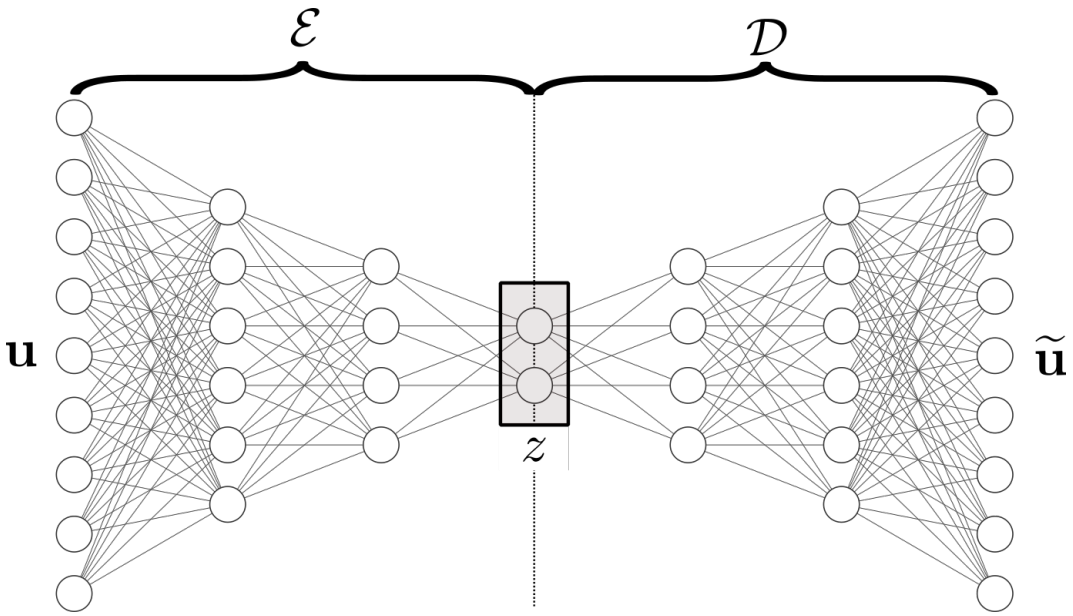


Figure 3.3: Illustration of an autoencoder used to learn the map from data samples $\boldsymbol{u}$ to a low dimensional embedding z

Figure 3.3 illustrates the proposed architecture. It can be trained by directly optimizing the Mean Squared Error (*MSE*) between the data points $\boldsymbol{u}$ and their reconstruction by the autoencoder $\widetilde{\boldsymbol{u}} = \mathcal{D}(\mathcal{E}(\boldsymbol{u}))$:

$$\mathcal{L} = \frac{1}{n_{\boldsymbol{u}}} \sum_{i=1}^{n_{\boldsymbol{u}}} \|\mathcal{D}(\mathcal{E}(\mathbf{u})) - \mathbf{u}\|_2^2. \tag{3.16}$$

We illustrate the method using the same three-dimensional dataset presented in Section 2.2.3. We show that the neural autoencoder is able to *disentangle* the data samples through its layers to learn a two-dimensional representation. Moreover, we see in figure 3.4 that the learned low-dimensional representation z is sufficient to reconstruct the original data manifold. Finally, the evolution of the data samples along the layers of the encoder network is also presented on the figure.

3.2 Learning the Dynamics

In the previous Chapter, we showed that data can be used to identify low-dimensional representation spaces for the solutions of numerical simulations, in order to reduce the computational costs of numerically simulating systems of interest.

Unfortunately, this is not a trivial matter, as the dynamics of the low dimensional embeddings extracted from data can be hard to identify. Indeed, rep-

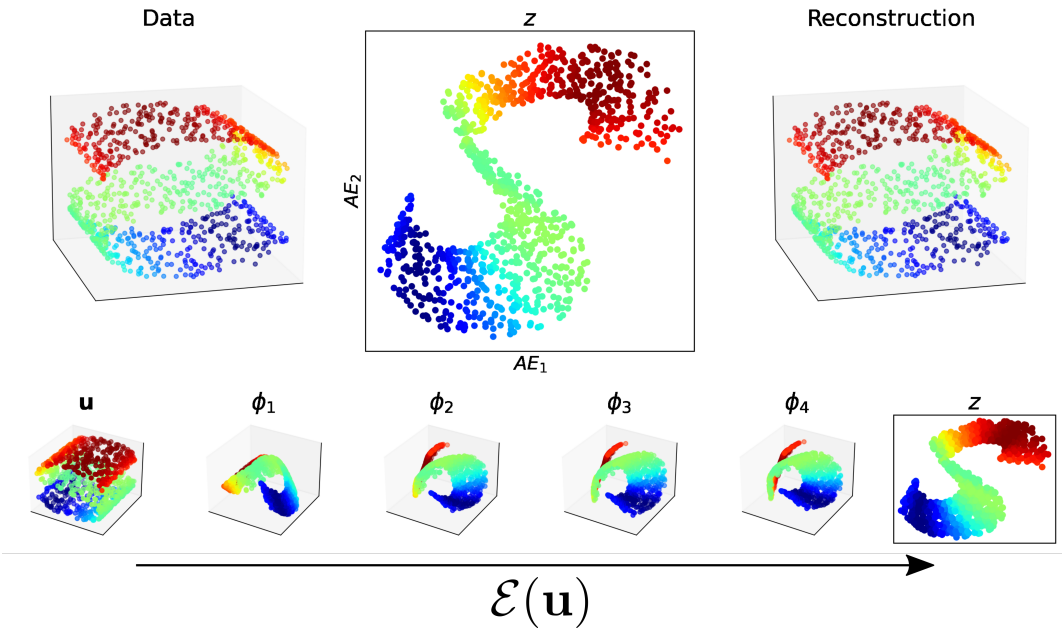


Figure 3.4: Performance of a neural Autoencoder on the reduction of a three-dimensional data manifold. *Top:* Two dimensional embedding and reconstruction of the data learned by the Autoencoder. *Bottom:* Evolution of the data samples projected on the leading principal components of the encoded data at each layer.

resentations unaware of the underlying manifold such as the Finite Elements Method were developed specifically so that they could be used to solve Partial Differential Equations, which are often the only model available for the accurate description of dynamical systems. Thus, changing the representation space to achieve dimensionality reduction complexifies the use of first-principle models, as reduced spaces are generally less suitable for the evaluation of PDEs.

In a more general setting, systems can not always be fully observed. For example, point-wise probes are often used for the experimental study of systems to measure the state at given spatiotemporal points. Similarly, in larger-scale applications such as weather modeling, the state is only measured sparsely over the world. These factors limit the applicability of standard PDE modeling methods and are sometimes addressed through expensive pre-processing steps using data assimilation methods (see for example Foures et al. [59]).

Both these scenarios are very similar from a dynamical modeling point of view as they are both related to the modeling of systems using only partial information, for which models are not readily available. A possible solution to avoid expensive data assimilation approaches combined with high dimensional models is to construct low dimensional models from the available system data directly.

3.2.1 Fully data-driven models

The following Paragraphs introduce some of the methods used for the direct identification of dynamics from data. There exists a large range of methods designed for this task. We chose some of the main methods used in reduced-order modeling contexts to introduce the principal challenges of the topic. Most notably, we want to emphasize the varying degrees of interpretability and theoretical justifications of each method, underlining that different methods designed for similar tasks can have strongly differing properties.

The Koopman operator and DMD

The theory of the Koopman operator provides an attractive theoretical framework for the construction of system dynamics, we present the main concepts below and refer the reader to Brunton et al. [169], Lin et al. [161], and Lin and Lu [160] for a complete and more formal derivation. The theory states that observables of the state of a system defined here as $g(\boldsymbol{u}) : \mathbb{R}^N \mapsto \mathbb{R}$ are advanced in time linearly by an operator called the Koopman operator ($\mathcal{K}$):

$$g(\boldsymbol{u}_{t+1}) = \mathcal{K}\boldsymbol{u}_t. \tag{3.17}$$

This formulation is advantageous, as the linear nature of the above system makes it extremely simple to simulate and analyze. Indeed, the time evolution of the observables $g(\boldsymbol{u}_t)$ can be computed without the expensive integration of an ODE, as the solution of a linear system. We note that when $g \equiv \psi_i$ with ψ_i an eigenfunction of the Koopman operator, we get:

$$\psi_i(\boldsymbol{u}_n) = \lambda_i^n \psi_i(\boldsymbol{u}). \tag{3.18}$$

With λ_i the associated eigenvalues and n the number of time steps taken. Thus, it appears that the eigenfunctions of the Koopman operator form an advantageous coordinate change, defining a space over which the dynamics are linear. The issue with this formulation is that the operating space of the Koopman operator is potentially infinite-dimensional. Meaning that for the approach to be computationally tractable, the Koopman operator has to be approximated and only part of its operating space considered.

For this purpose, it is practical to consider a set of observable functions $\{g_j(\boldsymbol{u})|j = 1, .., \infty\}$. It can be shown that under mild assumptions ([169] sec 2.3), each observable is expressed as a linear combination of the Koopman eigenfunctions:

$$g_j = \sum_{i=1}^{\infty} v_{j,i}\psi_i, \tag{3.19}$$

$$\implies g_{j,t+1} = \sum_{i=1}^{\infty} v_{j,i}\lambda_i\psi_i \tag{3.20}$$

A set of observables g_j can be constructed so that it forms a basis spanning the operating space of the Koopman operator, meaning that the Koopman eigenfunctions can be decomposed in terms of the observables $\psi_i = \sum_{k=1}^{\infty} w_{i,k}g_k$. Injecting this result into Eq.(3.18), a linear system for the dynamics of the observables g_j:

$$g_{j,t+1} = \sum_{i=1}^{\infty} \lambda_i v_{j,i} \sum_{k=1}^{\infty} w_{i,k}g_{k,t} \tag{3.21}$$

$$\implies \mathbf{g}_{t+1} = \mathbf{K}\mathbf{g}_t. \tag{3.22}$$

Where $\mathbf{g} = [g_1, g_2, \ldots]$ corresponds to the observables arranged into a vector, and $\mathbf{K}$ is a matrix representation of the Koopman operator. With this advantageous matrix formulation, truncations of the basis of observables g_j can be considered to approximate the dynamics in the above equation in a finite-dimensional operating space.

A simple approach to the approximation of these dynamics is called the Dynamic Mode Decomposition algorithm, proposed in Schmid and Sesterhenn [42]. The algorithm defines the linear dynamics directly over the state of the system so that $\mathbf{g}(\boldsymbol{u}) = \boldsymbol{u}$. This yields a linear system of equations:

$$\boldsymbol{u}_{t+1} = \mathbf{A}\boldsymbol{u}_t. \tag{3.23}$$

With $\mathbf{A}$ a truncation of the Koopman operator ($\mathbf{K}$). Note that this algorithm was already introduced in the context of dimensionality reduction in Section 2.2.2, indeed, the algorithm provides both a relevant basis of modes on which the state of the system can be expressed through the computation of the right eigenvectors of $\mathbf{A}$, and an advantageous formulation for the dynamics of the system. We note that this advantageous formulation can be exploited not only to forecast the state of a system in the future but also to analyze the system at hand, through the spectral properties of the operator $\mathbf{A}$ [55].

This formulation is sufficient for the identification of the dynamics when applied to linear or weakly nonlinear PDEs. In other cases, this approach loses its efficiency for the forecasting of the state of the system, although its interest for the identification of relevant dynamical modes remains. To adapt the method to nonlinear dynamics, approaches have been proposed to construct efficient observable bases $\mathbf{g}$ on which the dynamics can be represented linearly.

The Extended-DMD [77] uses dictionary learning methods to construct a basis of observables $\mathbf{g}(\boldsymbol{u}) = \left[f_1(\boldsymbol{u}), f_2(\boldsymbol{u}), \ldots, f_{N_f}(\boldsymbol{u})\right]$ with N_f generally well superior to the dimension of $\boldsymbol{u}$, on which the dynamics are represented linearly. Similarly, the Kernel-DMD [72] uses kernel methods to efficiently construct a basis of non-linear observables $\mathbf{g}$. Both these methods select a number of non-linear transformations of the state $\boldsymbol{u}$ to construct the observable sub-space on which the approximate Koopman operator acts.

More recently, neural networks have been used to learn this dictionary of observables [88, 90, 106, 114, 187]. These methods leverage the ability of neural networks to learn an efficient basis of functions on which the dynamics are linear. These extensions of the DMD method help address the issue of the truncation of the operating space of the Koopman operator. Indeed, although it is impossible to perfectly represent the dynamics of complex systems in a finite-dimensional space as a linear ODE, given enough data, neural networks are able to identify an efficient representation of the system, yielding a more accurate and interpretable representation of the dynamics.

Finally, we point out that in this Section, we have not discussed the topic of reduced order modeling using the Koopman operator. That is because the dimension of the coordinates vector $\mathbf{g}(\boldsymbol{u})$ is generally taken to be large, in accordance with the expectation that the Koopman operator is infinite-dimensional, which is in direct opposition with the idea of dimensionality reduction introduced in Section 2.2. However, we show in our work (see Chapter 6) that certain dynamical systems can be efficiently modeled in very low dimensional spaces as linear systems. Moreover, the theory of the Koopman operator introduced in this Section can be used to derive an *ansatz* for the dynamics of partially observed systems, which is shown to be critical to the construction of theoretically grounded dynamical systems. This particular topic is discussed in detail in chapter 6.

Sindy

The theory of the Koopman operator presents a very advantageous representation of the dynamics of a system. However, it is sometimes intractable because of the assumption of linearity of the dynamics and the dimension of the operating space of the Koopman operator. We introduce in this Section the Sparse Identification of Nonlinear Dynamics *SINDy* [79], as a method for the extraction of nonlinear dynamical models from data.

The SINDy algorithm is based on a simple dictionary-learning approach to approximate the derivative of the system at hand. First, the system snapshots $\boldsymbol{u}_i = \boldsymbol{u}(t_i)$ are assembled in a matrix $\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n_s}]$, then, a dictionary $\mathbf{D}(\mathbf{U}) \in \mathbb{R}^{d_\mathbf{D} \times d_{n_s}}$ of candidate functions is constructed:

$$\mathbf{D}(\mathbf{U}) = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_{n_s} \\ \boldsymbol{u}_1^2 & \boldsymbol{u}_2^2 & \dots & \boldsymbol{u}_{n_s}^2 \\ \sin(\boldsymbol{u}_1) & \sin(\boldsymbol{u}_2) & \dots & \sin(\boldsymbol{u}_{n_s}) \\ \dots & \dots & \dots & \dots \end{bmatrix}. \tag{3.24}$$

The true value of the time derivative of each snapshot $\boldsymbol{u}_i$ is then computed, either from the governing equations if they are known, or approximated through finite differences to obtain the matrix $\dot{\mathbf{U}} = [\dot{\boldsymbol{u}}_1, \dot{\boldsymbol{u}}_2, \dots, \dot{\boldsymbol{u}}_{n_s}]$. To obtain a representation of the dynamics in the dictionary of functions, a matrix of coefficients $\mathbf{C} \in \mathbb{R}^{d_{\boldsymbol{u}} \times d_{\mathbf{D}}}$ to be learned is introduced, so that the time derivative is expressed as follows:

$$\dot{\mathbf{U}} = \mathbf{C}\mathbf{D}(\mathbf{U}). \tag{3.25}$$

The goal of the method is to identify a sparse representation of the dynamics in the dictionary of functions $\mathbf{D}$. Thus, the following optimization problem is solved:

$$\mathbf{C} = \arg\min_{\hat{C}} \|\dot{\mathbf{U}} - \hat{\mathbf{C}}\mathbf{D}(\mathbf{U})\| + \lambda \|\hat{\mathbf{C}}\|_1. \tag{3.26}$$

With λ a coefficient controlling the importance of the sparsity-promoting term in the above loss. Solving the above optimization problem yields a matrix of coefficients $\mathbf{C}$, whose rows correspond to the coefficients of a linear combination of the functions in $\mathbf{D}(\mathbf{U})$ that approximates the dynamics of each dimension of the state $\boldsymbol{u}$.

Note that the above procedure has few theoretical justifications as it only trains a model to predict the value of the time derivative extracted from data, which can be highly sensitive to noise and lead to inaccurate predictions. Moreover, it has two major limitations that are intrinsic to dictionary learning. First, the choice of functions that make up the dictionary must be made *a-priori*, meaning there is no guarantee that the dictionary is expressive enough for the dynamics to be efficiently represented. The more important issue, however, is that the number of functions in the dictionary scales exponentially with the dimension of the state $\boldsymbol{u}$. Indeed, combinations of each dimension of the state must be added to the dictionary to capture systems as simple as the Lorenz attractor. This last factor makes it hard to construct a tractable dictionary that retains expressivity for relatively high dimensional states.

The approach does however have one major advantage which is its interpretability. Indeed, once trained, the *SINDy* model can be easily analyzed. Works such as Loiseau and Brunton [105] and Callaham, Brunton, and Loiseau [155] have shown that this interpretable aspect can be leveraged into physical constraints on the structure of the model, or used to gain insights into the behavior of the system under study. Similar ideas have also been used to derive interpretable nonlinear closures, as in Kalur et al. [197].

Recurrent Neural Networks

The Koopman operator, and to a lesser extent, the SINDy algorithm, both provide principled ways to identify dynamical models from data that have the advantage of being interpretable. Unfortunately, we have discussed how these models are limited, either by their lack of expressivity, or by their computational costs. We introduce in this Section Recurrent Neural Networks (*RNN*s) as a way to extract dynamical models from data, with very little *a-priori* constraints on the structure of the model.

Recurrent Neural Networks are general neural network architectures that have loops in the connectivity graph. They have been intensively used to handle sequential data. Two main RNN architectures have emerged: the Long Short Term Memory (*LSTM*) proposed in 1995 [23], and the more recent and simpler Gated Recurrent Unit *GRU* proposed in 2014 [56]. Both these approaches have been used extensively in most application fields of Deep Learning. In this Section, we focus on the LSTM network as it has been used in works that are relevant to this thesis [201, 184].

The LSTM architecture was proposed as a model able to handle temporal dependencies in time series and retain information along its trajectory. Thus, it is able to capture non-markovian effects in the dynamics of a system. To do so, the LSTM advances an internal state in time, which can be seen as a memory, using a series of *gating* mechanisms. We introduce the equations for the LSTM from a dynamical point of view, where the model is trained to predict the next timestep $\boldsymbol{u}_{t+1}$ based on the current state $\boldsymbol{u}_t$, the memory $\mathbf{c}_t$ and potential exogenous inputs $\mathbf{x}_t$. The LSTM does so through the following equations:

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_t + b_f), \tag{3.27}$$
$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_t + b_i), \tag{3.28}$$
$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_t + b_o), \tag{3.29}$$
$$\widetilde{\mathbf{c}}_{t+1} = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_t + b_c), \tag{3.30}$$
$$\mathbf{c}_{t+1} = \mathbf{f}_t \odot c_t + \mathbf{i}_t \odot \widetilde{\mathbf{c}}_{t+1}, \tag{3.31}$$
$$\boldsymbol{u}_{t+1} = \mathbf{o}_t \odot \tanh(\mathbf{c}_{t+1}) \tag{3.32}$$

where $W_\_$ and $U_\_$ are weight matrices and $b_\_$ bias vectors, to be learned, while σ is the sigmoid function. The logic behind the equations is based on *gating*, *i.e.* the three *gates* $\mathbf{f}, \mathbf{i}, \mathbf{o}$, bounded between 0 and 1, are used to select the information in the various quantities handled by the LSTM to be passed forward in time. We see that the *forget* gate $\mathbf{f}_t$ is used to select the information in the previous memory state $\mathbf{c}_t$ to be passed through; The *input* gate i_t meanwhile computes the amount of the new tentative memory $\widetilde{\mathbf{c}}$ to be passed in memory; And the *output* gate o_t computes the next state $\boldsymbol{u}_{t+1}$ from the memory $\mathbf{c}$. Note that this is a simplified version of the architecture, in practice, the model outputs a hidden

state h_t from which the output $\boldsymbol{u}_t$ is computed. Moreover, multiple versions of this architecture are generally stacked to increase expressivity, so that the output of each LSTM cell is fed as input to the cell above it.

As discussed, this complex architecture is able to extract dynamic information from time series data. Its main advantage is its expressivity and scalability. Indeed, it is much more expressive than a simple linear model or a relatively large dictionary of functions. It is also able to transparently handle non-Markovian effects in the model which is critical in certain reduced-order modeling applications (see Chapter 4). It should however be noted that LSTM networks have to be used as a *black box*. Indeed, it is extremely difficult to gain insights into the model once it is trained, and there is no guarantee that it will remain stable over long integration periods. Because of these drawbacks, their usability in critical industrial applications has remained limited. They constitute however an example of the advantage of Neural Networks over other approaches in terms of expressivity and tractability.

As mentioned at the beginning of this Section, the above methods are not an exhaustive list of the fully data-driven methods that have been proposed to forecast the dynamics of physical systems. We cite, for example, approaches based on cluster models [63], on graph neural networks [166], on reservoir computing [128, 179, 180] or more recently, approaches based on transformers [193], which use machine learning to extract dynamical models from data directly. However, with the three above methods, we hope to have illustrated the breadth of the field of data-driven dynamical modeling. We have tried to underline the tradeoff between the interpretability and guarantees embedded in the model architecture and its expressivity. In fact, most of the work presented in this thesis is concerned with this specific point, as constructing expressive models that retain certain properties in terms of interpretability and theoretical validity is still very much an open problem. In the following Sections, we discuss this topic in more detail, introducing a range of methods developed to *hybridize* fully data-driven deep learning approaches with physical concepts. We then discuss the way reduced order models can sometimes be constructed directly from the governing equations to avoid using fully data-driven models.

3.2.2 On the importance of Hybridisation

As mentioned above, the construction of reliable and interpretable dynamical models is at the center of this thesis. We have shown in the previous Sections that fully data-driven methods can be used to obtain models that capture the dynamics embedded in system data. However, such approaches are limited by several factors:

- **Expressivity:** The choice of model to be trained can have limited expressivity and cannot fully capture the complexities of certain dynamical sys-

tems.

- **Generality:** The validity range of fully data-driven models is a central topic in Machine Learning research as it is well recognized that the performance of such models rapidly degrades when getting outside their training conditions. A large body of work has been dedicated to the issue, and despite the development of a wide range of methods to improve the generalization performance of data-driven models (regularization[5], ensembling[27], dropout[65] ...), this topic still constitute a major limiting factor to the use of Machine Learning approaches in critical applications.
- **Data availability:** Because of their complete reliance on the availability of data, fully data-driven models require large amounts of data to gain an acceptable level of accuracy. In industrial applications, data is very often scarce, as systems can be expensive to simulate and even more expensive to observe experimentally under various conditions (e.g., failures).

Various steps can be taken to address these limitations, such as embedding physical constraints in the structure of the model, or retaining part of the full-order PDEs that describe the dynamics of the system in the model. These ideas are related to the topic of inductive biases discussed in Section 3.1.4 as their goal is to embed the model with constraints that it should verify to avoid having to learn them, simplify training and construct architectures that better fit the task at hand.

3.2.3 Deep Learning and physics

The above considerations have led to a large body of work on the topic of hybridization between Deep Learning and physical models. We describe in this Section a few methods that have been proposed in the general context of hybridization between Deep Learning and PDE simulation.

Implicit representation

Implicit methods, also called *mesh-free* methods, propose to represent the state of a system as a neural network, *i.e.* the state of the system $u(x,t)$ is not represented as a linear combination of a number of basis functions as in the Finite Elements method, but rather as the value of a function chosen to be a neural network. These methods have the advantage of bypassing the issues inherent to high dimensional discretizations (see sec 2.1), but also provide a degree of *hybridization* as they leverage the governing equations to train the neural network.

Various approaches following this idea have been proposed, we cite for example Berg and Nyström [97] who used fully connected layers to approximate

the solution of simulation problems on complex geometries and Sirignano and Spiliopoulos [110] who proved that multi-layer perceptrons can approximate the solution of a large class of PDEs, provided they are *wide* enough. We also summarise below the well known *physics informed neural networks* approach (*PINNs*, [109, 92, 93, 124]). This method is based on the ideas presented above, where a neural network $u(x,t;\theta)$ is trained to solve a PDE problem:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \mathbf{G}(u(x,t),t),\ x \in \Omega, t \in [0,T], \\ u(x,0) &= u_0(x), \\ f_{\partial\Omega}(u(x,t)) &= 0,\ \forall x \in \partial\Omega. \end{aligned} \tag{3.33}$$

Where u_0 is the initial condition of the problem and $f_{\partial\Omega}(u(x,t))$ represents the boundary conditions imposed at the boundary $\partial\Omega$ of the computational domain Ω. In the *PINNs* approach, the neural network $u(x,t;\theta)$ is simply trained to verify the above equations through gradient descent, by minimizing the following loss:

$$\mathcal{L} = \frac{1}{N_\Omega}\sum_{i=1}^{N_\Omega} \|\frac{\partial u_i}{\partial t} - \mathbf{G}(u_i,t_i)\| + \frac{1}{N_{\partial\Omega}}\sum_{i=1}^{N_{\partial\Omega}} \|f_{\partial\Omega}(u_i)\| + \frac{1}{N_0}\sum_{i=1}^{N_0} \|u(x_i,0) - u_0(x_i)\|. \tag{3.34}$$

Where each term in the loss enforces one of the constraints in Eq.(3.33) and the notations $N\_$ correspond to the pre-defined number of points on which each term is evaluated. Minimizing this loss means that the model is trained to respect the physical conditions governing the problem, rather than extract them from data. This method can be used to directly forecast states using the trained neural network $u(x,t)$, but also to reconstruct a system's state from sparse data when used to solve inverse problems [174, 178, 59], or fit model parameters. We note that implicit representation methods have also seen some use in the context of reduced-order modeling [171].

Unfortunately, these methods have significant drawbacks, mainly their computational cost which can sometimes be on par with the cost of directly solving the problem in a standard PDE solver. They are also hindered by their generalization capabilities, as balancing the tradeoff between the expressivity of the neural network $u(\cdot;\theta)$ and avoiding overfitting can be complicated.

Neural Operators

Very similar to the above methods, Neural operators propose to learn implicit representations of the operator that solves a given parametric problem, yielding a continuous solution to parametric PDE problems. To illustrate the approach, we briefly introduce the *DeepONet* [162], which proposed one of the

first neural operator learning approaches. The goal of the method is to learn the map $\mathcal{G} : \mathcal{P} \mapsto \mathcal{S}$ between $\mathcal{P}$ the space of functions representing the parameters of the PDE problem to be solved, and $\mathcal{S}$ the space of its solutions. The map is parameterized using a neural network $\mathbf{G}$ so that:

$$\mathbf{G}_\theta[p](x,t) = s(x,t). \tag{3.35}$$

Where (x,t) are the spatio-temporal coordinates and p the parameters. To do so, the *neural operator* $\mathbf{G}_\theta$ is separated in a number of so-called *branch*-networks $b_j(\cdot;\theta)$ and a *trunk*-network $t(x,t;\theta)$. The *branch*-networks b_j handle the parametric nature of the PDE problem, while the *trunk*-network t_j learns the spatiotemporal dependencies of the solution. To ensure that the dimension of the input of the network is coherent between parametric cases, a discretized representation of the function p is generally used, so that the network takes the following final form:

$$\mathbf{G}_\theta[p](x,t) = \sum_{j=1}^{N_b} b_j\left(p(\eta_1),\dots,p(\eta_F)\right);\theta) t_j(x,t;\theta). \tag{3.36}$$

With η_i the discretization points used to represent the parameters p. Lu et al. [162] proved that the above formulation could approximate the map $\mathcal{G}$ to any desired degree of precision. The method can be trained by gathering data triplets $((x,t),p,s(x,t))$ so that the following loss is minimised:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \|s(x_i,t_i) - \mathbf{G}_\theta[p_i](x_i,t_i)\|. \tag{3.37}$$

This neural operator has the advantage of structurally accounting for the parametric nature of the dynamical problem at hand by separating the parametric and spatiotemporal parts of the solution. This alleviates the need to retrain the architecture when generalizing to new parametric conditions. Extensions of the approach have been proposed such as Kaltenbach, Perdikaris, and Koutsourelakis [188] and Li et al. [138], and most notably, the Fourier Neural Operator [137].

Despite these advances, the efficiency of the approach still remains limited due in part to the complexity of learning the results of long-range integral problems that result from the resolution of PDEs for arbitrary integration times. We note however that these models constitute an interesting proposal for the embedding of inductive biases specific to the nature of parametric PDE problems in the architecture of neural networks.

Stability constraints

Various proposals have also been made to embed neural networks with stability constraints, starting with Hamiltonian Neural Networks (*HNN*) which were

developed to model the dynamics of conservative systems [118]. The approach proposes a very simple model architecture that ensures the stability of the resulting model. This is achieved by leveraging the structure of Hamiltonian systems, which are based on the partial derivatives of a scalar function $\mathbf{H}(\mathbf{p}, \mathbf{q})$:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} &= \frac{\partial \mathbf{H}}{\partial \mathbf{p}}, \\ \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} &= -\frac{\partial \mathbf{H}}{\partial \mathbf{q}}, \end{aligned} \tag{3.38}$$

where $(\mathbf{p}, \mathbf{q})$ are the generalized coordinates of the system. The HNN approach proposes to parametrize the Hamiltonian function $\mathbf{H}$ of a conservative system with unknown dynamics as a neural network. Indeed, the conservative nature of the above system is independent of the choice of the function $\mathbf{H}_\theta$, thus, the above model is guaranteed to be stable, even outside its training conditions. Similar ideas were proposed to ensure the invertibility of neural networks, Behrmann, Duvenaud, and Jacobsen [113] proposed to constrain the Lipschitz constant of the weight matrices of a neural network to ensure its invertibility, which is equivalent to ensuring the conservation of information through each layer of the network, while Haber and Ruthotto [86] proposed novel parameterization schemes for weight matrices based on Hamiltonian mechanics to ensure the stability of the model.

Of course, the Hamiltonian framework is often found to be excessively restrictive for the modeling of physical problems, as most dynamical systems are dissipative, and often forced by exogenous factors, such as an inflow boundary condition in the case of a fluid problem. However, parameterization schemes were proposed to ensure the stability of neural dynamical models. For example, N. B. Erichson and Mahoney [121] and Pan and Duraisamy [142] proposed to use dissipative parameterizations to enforce the stability of trainable linear operators.

Neural ODEs

We introduce a last neural modeling approach, called Neural ODEs [99], which is used extensively in our work. The method borrows concepts from dynamical systems to construct continuous (or infinite) depth neural networks. These methods are derived from the Residual Network [82] architecture, which is a slight modification of the multi layer perceptron introduced in Section 3.1.1. A residual network $\mathbf{\Phi}_\theta(x)$ learns a transformation of an input x as follows:

$$\begin{aligned} \Phi_\theta(x) &= \phi_{N,\theta} \circ \ldots \circ \phi_{3,\theta} \circ \phi_{2,\theta} \circ \phi_{1,\theta}(x), \\ \phi_{i,\theta}(x) &= x + \psi_{i,\theta}, \end{aligned} \tag{3.39}$$

where $\psi_{i,\theta}$ are trainable transformations, such as the linear layer defined in Section 3.1.1. This approach was proposed to handle increasingly deep network architectures and quickly became state of the art, notably for image classification tasks. Observing that the dimension of the layers of the residual network in equation 3.39 is constant and equal to the dimension of the input x, Eq.(3.39) can be viewed as the Euler integration scheme of a dynamical system, where the layer ϕ_i corresponds to the i_{th} time step of the integration. Chen et al. [99] proposed to directly learn the dynamics of the transformation so that:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= f(x;\theta), \\ \mathbf{\Phi}(x_0) &= x_0 + \int_0^T f(x_t;\theta)dt, \end{aligned} \tag{3.40}$$

where T is a pre-determined integration horizon, f is a trainable transformation, defined as a neural network, and the output of the network, x_T, results from the integration of these trainable dynamics. Because the dynamics f can be trained to be arbitrarily stiff, the number of integration steps, which corresponds to the number of layers in a Residual Network, can be as high as required by the task at hand. Similar to standard neural networks, Neural ODEs are trained to minimize a loss function $\mathcal{L}(x_T)$, however, the discrete backpropagation algorithm introduced in Section 3.1.2 is ill-suited for this task. Instead, Adjoint backpropagation, an adaptation of the backprop algorithm to time continuous problems, is used.

Adjoint Backpropagation

The adjoint backpropagation algorithm on which Neural ODEs are based is used to solve problems of the following form[3]:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{L}(x(T)) \\ \text{s.t.} \quad & \frac{\mathrm{d}x}{\mathrm{d}t} - f(x;\theta) = 0 \\ & x(0) = x_0. \end{aligned} \tag{3.41}$$

These problems are generally solved through gradient descent, which requires evaluation of the following gradient:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} = \frac{\partial \mathcal{L}}{\partial x}\frac{\partial x}{\partial \theta}\Big|_T. \tag{3.42}$$

[3]This specific criterion and constraints choice correspond to the training objective of Neural ODEs. The adjoint backpropagation algorithm can be used to solve more complex problems; however, we chose to restrict the scope to this formulation to clarify the derivation.

Because the derivative of $\dot{x}(t) = f(x;\theta)$ is parameterised by θ, there is an implicit relation $x = x(t;\theta)$. The estimation of the sensitivity $\frac{\partial x}{\partial \theta}|_T$ requires the consideration of the impact of θ on the whole time-integration, from $t = 0$ to $t = T$. The adjoint backpropagation is used to avoid computing this term, which is done by evaluating the sensitivity of the following Lagrangian:

$$\mathcal{J} = \mathcal{L} + \int_0^T \mu(t)(\dot{x}(t) - f(x;\theta))\mathrm{d}t \tag{3.43}$$

$$\dot{x}(t) = f(x;\theta) \implies \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta}. \tag{3.44}$$

The vector of Lagrangian multipliers μ is a function of time. Distributing the product in the integral and integrating the first term by parts leads to an expression where the sensitivity $\frac{\partial x}{\partial \theta}\Big|_T$ can be isolated:

$$\mathcal{J} = \mathcal{L} + \Big[\mu x\Big]_0^T - \int_0^T \dot{\mu} x(t) + \mu(t) f(x;\theta)\mathrm{d}t \tag{3.45}$$

$$\implies \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta} = \Big(\frac{\partial \mathcal{L}}{\partial x} + \mu(T)\Big)\frac{\partial x}{\partial \theta}\Big|_T - \mu(0)\cancelto{0}{\frac{\partial x}{\partial \theta}\Big|_0} - \int_0^T \Big(\dot{\mu} + \mu(t)\frac{\partial f}{\partial x}\Big)\frac{\partial x}{\partial \theta}\Big|_t + \mu(t)\frac{\partial f}{\partial \theta}\Big|_{x=x(t)}\mathrm{d}t. \tag{3.46}$$

A so-called adjoint equation is then obtained to avoid having to solve for the sensitivity $\frac{\partial x}{\partial \theta}$. Enforcing a vanishing variation of the Lagrangian *wrt* x at optimality yields:

$$\begin{aligned} \frac{\mathrm{d}\mu}{\mathrm{d}t} &= -\mu\frac{\partial f}{\partial x}\Big|_t \\ \mu(T) &= -\frac{\partial \mathcal{L}}{\partial x}\Big|_T. \end{aligned} \tag{3.47}$$

Solving the adjoint equation (3.47) for the values of $\mu(t)$ leads most of the terms in Equation (3.46) to vanish, so that:

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta} = -\int_0^T \mu(t)\frac{\partial f}{\partial \theta}\Big|_{x=x(t)}\mathrm{d}t. \tag{3.48}$$

Both this integral and the adjoint equation can be easily evaluated if $\frac{\mathrm{d}x}{\mathrm{d}t}$ is approximated by a neural network as the required vector-Jacobian products $\mu\frac{\partial f}{\partial x}$ and $\mu\frac{\partial f}{\partial \theta}$ can be easily computed in any deep learning framework.

Adaptive checkpoint adjoint

The original NeuralODE paper [99] proposed to integrate forward in time to obtain the initial condition of the adjoint equation evaluated from $x(T)$, while discarding intermediate values $x(t)$. This choice was made with the goal of reducing the memory footprint of the method. However, discarding the intermediate

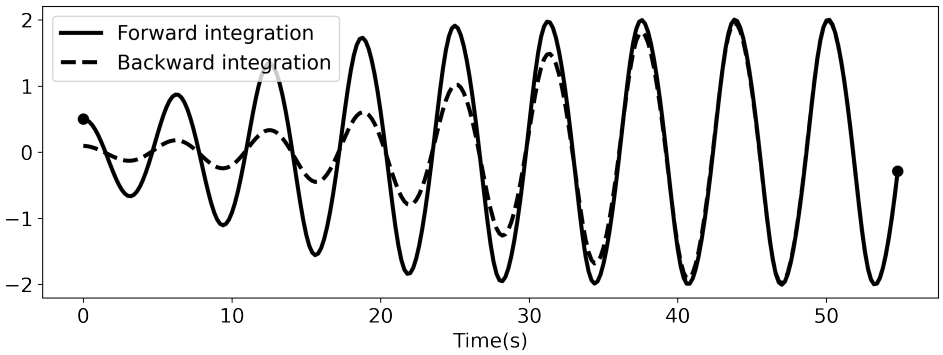


Figure 3.5: Forward and backward time integration of the Van der Pol oscillator. Numerical errors during the integration lead to the divergence of the two trajectories which should be identical.

time-steps $x(t)$ means that they have to be recomputed during backpropagation to evaluate the adjoint ODE (3.47), as well as the gradient integral (3.48). This can be done as a single backward-in-time integration by concatenating the different quantities $(x, \mu, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta})$ in a single state vector:

$$z = \left[x, \mu, \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\theta}\right] \tag{3.49}$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \left[f(x;\theta), -\mu\frac{\partial f}{\partial x}, -\mu\frac{\partial f}{\partial \theta}\right] \tag{3.50}$$

$$z(T) = \left[x(T), -\frac{\partial \mathcal{L}}{\partial x}\Big|_T, \frac{\partial \mathcal{L}}{\partial x}\frac{\partial f}{\partial \theta}\Big|_T\right]. \tag{3.51}$$

Not only does this increase the computational cost of the method, but it can also lead to erroneous gradients, as numerical errors during integration can lead to differences between the forward and backward trajectories for $x(t)$. This was observed in Zhuang et al. [150] and is illustrated in Figure 3.5, which shows that, despite using the same parameters, the forward and backward time trajectories diverge due to numerical errors. To address this issue, one can use the Adaptive Checkpoint Adjoint method. This method retains the intermediate integration steps $(t, x(t))$ and simply evaluates the integrals for the adjoint μ and the gradient $\frac{\mathrm{d}J}{\mathrm{d}\theta}$ at the time steps selected during the forward integration, using the forward trajectory $x(t)$.

Although this increases the memory footprint of the method, it limits the computational cost of the backward pass, as the time steps are already selected, and the trajectory $x(t)$ does not have to be integrated another time. Most importantly, this limits the numerical errors introduced by the integration schemes, which can have a significant impact on the training, especially in the case of chaotic dynamics.

3.2.4 Summary

The approaches presented above all stand as various forms of hybridization. Methods such as the physics informed neural networks, or the neural operator, propose to leverage the properties of neural networks to efficiently solve PDE problems. Similarly, the methods described in subsection 3.2.3 as well as the NeuralODE method propose to embed neural network architectures with previously established properties such as stability or time continuity. These properties can both help guarantee the results of a trained model and simplify training.

3.3 The POD Galerkin method

Retaining Physics through the Galerkin Projection

In this Section, we come back to the topic of reduced order modeling, and we show that while the various methods listed in the previous Section can be used to adapt Deep Learning approaches to physical modeling, it is possible to directly retain part of the governing equations when designing a reduced order model. Thus retaining a dynamical *basis* that can then be hybridized with data-driven approaches to create stable and accurate reduced order models. To this end, the Galerkin Projection method is introduced as an efficient reduced-order modeling method that leverages the governing equations of the problem under study to avoid sole reliance on data. We also refer the reader to Holmes et al. [48] and Lassila et al. [64] for additional discussions on the topic.

Linear dimensionality reduction methods such as POD have a significant advantage for the construction of hybrid reduced-order models. Indeed, they can be used in combination with the Galerkin projection method to retain part of the dynamics in the reduced space. The method starts from a discretized full-order model as follows:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \mathbf{F}(t, \boldsymbol{u}) \qquad (N\, equations), \tag{3.52}$$

where $\boldsymbol{u} \in \mathbb{R}^N$ is the high dimensional state of the system discretized in the chosen full-order representation space (it could for example be the coefficients of a Finite Elements basis), and $\mathbf{F}$ is the discretized full order model, for example, the discretized incompressible Navier Stokes equations, used for the simulation of simple fluid flows. Starting from this high dimensional system of equations and a number of pre-acquired simulation snapshots $\boldsymbol{u}(t_i)$, a reduced POD basis of principal modes $\mathbf{V_r} \in \mathbb{R}^{N \times r}$, $r \ll N$ is identified following the procedure described in Section 2.2.2. With this basis, an approximate solution can be reconstructed:

$$\widetilde{\boldsymbol{u}}(t) = \mathbf{V_r}\boldsymbol{a}(t) \tag{3.53}$$

$$\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}\|_2 \ll \|\boldsymbol{u}\|_2 \tag{3.54}$$

where $\boldsymbol{a}(t) \in \mathbb{R}^r$ is the reduced coordinate vector of the state in the POD basis, whose value is obtained from the full order state through projection $\boldsymbol{a}(t) = \mathbf{V_r}^\mathsf{T}\boldsymbol{u}(t)$. Similarly, Eq.(3.52) can be projected to obtain a system of dynamical equations for the coordinates $\boldsymbol{a}(t)$:

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \mathbf{V_r}^\mathsf{T}\mathbf{F}(t, \boldsymbol{u}) \qquad (r\ equations). \tag{3.55}$$

The projection above yields a system of r ODEs that exactly control the time evolution of the reduced coordinates $\boldsymbol{a}$. However, it is unusable in a reduced order modeling context as it assumes knowledge of the full order state $\boldsymbol{u}(t)$ during the simulation. To close the above system in $\boldsymbol{a}$, the approximate state $\boldsymbol{u} \approx \mathbf{V_r}\boldsymbol{a}$ is injected in Eq.(3.55) to evaluate the dynamics, introducing approximation errors in the model:

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} \approx \mathbf{V_r}^\mathsf{T}\mathbf{F}(t, \mathbf{V_r}\boldsymbol{a}) \qquad (r\ equations). \tag{3.56}$$

The above procedure is called a Galerkin projection and is similar to the construction of weak forms in the Finite Elements Method [50]. In next Section, we discuss the way this reduced model can be constructed practically and efficiently, by exploiting the linearity of the derivative operator.

Computational efficiency

First principle models are generally expressed as partial differential equations. Thus, the linear nature of the derivative operator can be leveraged to efficiently reduce the full order model $\mathbf{F}$ in Eq.(3.56) into a number of low dimensional tensorial operations. To give an example, we consider the incompressible Navier-Stokes Equations which control the behavior of the velocity and pressure fields $u(x,t)$ and $p(x,t)$ of a fluid flow at reasonably low Reynolds numbers (Re):

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\nabla p - (u \cdot \nabla)u + \frac{1}{Re}\nabla^2 u \\ \nabla \cdot u &= 0 \end{aligned} \tag{3.57}$$

These equations can be discretized and projected on a Finite Element basis following the procedure introduced in Section 2.1 to obtain a system of equations for the dynamics of the FE coefficient vectors $\boldsymbol{u}(t)$ and $\mathbf{p}(t)$:

$$\begin{aligned} &\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = -A\mathbf{p} - (\boldsymbol{u}^\mathsf{T} \cdot Q)\boldsymbol{u} + \frac{1}{Re}B\boldsymbol{u}, \\ &C\boldsymbol{u} = 0. \end{aligned} \tag{3.58}$$

Where A, B, C are high dimensional matrices resulting from the discretization of the differential operators. Indeed, the gradient, Laplacian and divergence operators in the Navier-Stokes equations are all linear and naturally result in discretized linear operators on the state of the system. Finally, the operator Q is a three dimensional tensor representing the quadratic advection term $(u \cdot \nabla)u$ in the NS equations. Note that all these operators are extremely sparse as they only model local interactions in the flow field, meaning that the evaluation of the above expression can be optimized numerically to improve performance.

Projecting this equation on a POD basis $\mathbf{V_r}$ and injecting the approximate state $\boldsymbol{u} \approx \mathbf{V_r}\boldsymbol{a}$, a system of r ODEs is obtained:

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \frac{1}{Re}\mathbf{V_r}^{\mathsf{T}} B \mathbf{V_r}\boldsymbol{a} - \mathbf{V_r}^{T}\boldsymbol{a}^{\mathsf{T}}(\mathbf{V_r} \cdot Q)\mathbf{V_r}^{\mathsf{T}}\boldsymbol{a}. \tag{3.59}$$

Note that both the mass conservation condition ($C\boldsymbol{u} = 0$) and the pressure term (Ap) have been dropped from the equation. This is due to the fact that the reconstructed flow field $\mathbf{V_r}\boldsymbol{a}$ is divergence-free by construction as the POD modes V_r all respect mass conservation (see Section 2.2.2), thus, the second equation is always verified. The pressure term can be dropped because the pressure field relates to the local compression or stretching of the flow, which is null for a divergence-free field, thus, the pressure field is constant and its gradient is null.

The advantage of this tensorial reduced order model is that a large part of the operations can be computed prior to the simulation of the ROM to obtain the filtered operators $\widetilde{B} \in \mathbb{R}^{r\times r}$ and $\widetilde{Q} \in \mathbb{R}^{r\times r\times r}$, yielding an efficient ROM that is easily evaluated through a few *r-dimensional* vector-matrix multiplications:

$$\widetilde{B} = \mathbf{V_r}^{\mathsf{T}} B \mathbf{V_r}, \tag{3.60}$$

$$\widetilde{Q} = \mathbf{V_r}^{\mathsf{T}}(\mathbf{V_r} \cdot Q)\mathbf{V_r}, \tag{3.61}$$

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \frac{1}{Re}\widetilde{B}\boldsymbol{a} - \boldsymbol{a}^{\mathsf{T}}\widetilde{Q}\boldsymbol{a}. \tag{3.62}$$

This example of the incompressible Navier Stokes equations shows that linear reduction methods are very advantageous as they allow for the construction of reduced dynamical models entirely based on the governing equations. It should be noted that while the Galerkin Projection method works well in the general setting of Partial Differential Equations, it can become inefficient for the reduction of strongly nonlinear equations. An example would be the case of chemical source terms that are generally modeled using Arrhenius laws, for which the above procedure of pre-assembling low-dimensional dynamical terms would not be applicable. For these cases, methods such as the discrete empirical interpolation method [43] have been proposed, relying on the evaluation of the irreducible dynamics on a low number of well-chosen spatial points

to construct the low-order representation of the nonlinear terms at each integration time.

Limitations of the Galerkin Projection

The Galerkin Projection can efficiently exploit linear dimensionality reduction and pre-existing equations to construct a dynamical model for the low dimensional system representation. This comes at the cost of approximation errors as the model is only exact when the projected dynamics are evaluated from the full-order model:

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \mathbf{V_r}^\mathsf{T}\mathbf{F}(t, \boldsymbol{u}). \tag{3.63}$$

Because the state of the system is only partially resolved in the reduced order setting, the above expression cannot be computed directly. Using a Taylor expansion, the dynamics can be separated into resolved and unresolved parts:

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \mathbf{V_r}^\mathsf{T}\mathbf{F}(t, \widetilde{\boldsymbol{u}} + \underbrace{\boldsymbol{u} - \widetilde{\boldsymbol{u}}}_{\textit{unresolved part}}), \tag{3.64}$$

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = \mathbf{V_r}^\mathsf{T}\mathbf{F}(t, \widetilde{\boldsymbol{u}}) + \underbrace{\mathbf{V_r}^\mathsf{T}\frac{\partial \mathbf{F}}{\partial \boldsymbol{u}}\Big|_{\widetilde{\boldsymbol{u}}}(\boldsymbol{u} - \widetilde{\boldsymbol{u}}) + \dots}_{\epsilon} \tag{3.65}$$

This expansion shows that the error term ϵ in the reduced order model directly depends on the unresolved part of the state ($\boldsymbol{u} - \widetilde{\boldsymbol{u}}$). Despite the relatively low magnitude of this unresolved part (see Eq.(3.54)), this term has a significant impact on the quality of the model. Indeed, the errors on the dynamics compound over time, leading to inaccurate trajectories of the reduced coordinates $\boldsymbol{a}$. This topic has been extensively studied, with the development of more advanced projection schemes to account for the residual ϵ during the integration of the model. Choi and Carlberg [115] and Carlberg, Barone, and Antil [84] provide a thorough study of the stability and optimality of the Galerkin Projection, as well as other projection schemes, showing that some of the limitations of the Galerkin projection can be alleviated, at the cost of increased computational overhead.

In this Section, we have shown that linear reduction methods can be coupled with well-established PDE projection methods to construct reduced-order models from the governing equations. Despite providing a model unconcerned with the various issues inherent to fully data-driven models (see Section 3.2.2), they are limited by the reconstruction errors associated with the use of incomplete linear bases. Indeed, their errors are directly related to the complement of the projection on the linear basis $\widetilde{\boldsymbol{u}} - \boldsymbol{u}$.

This concludes this Section on the construction of dynamical models when governing equations are unavailable, or can not directly be used. We have shown that a range of methods are available, with various degrees of accuracy and interpretability. We have discussed the issues that come with directly learning models from data as well as the interest in retaining the original equations as much as possible when deriving models for a low-dimensional representation of a system.

3.4 Conclusion

This Section introduced the topics at the center of the thesis. We have shown that there is a continuum of modeling approaches between standard PDE simulation and Deep Learning for dynamical models.

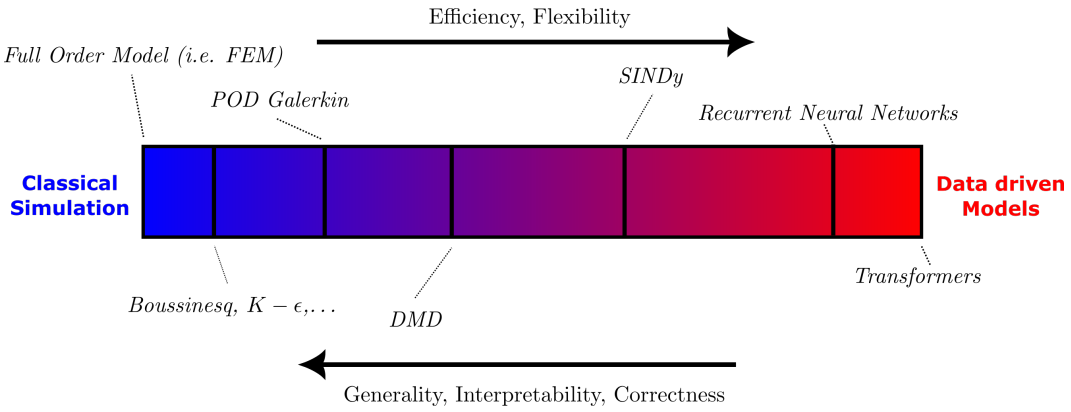


Figure 3.6: *Hybridization spectrum* between full order methods and fully data-driven methods. NB: The annotated methods only constitute a sample of the various approaches relevant to the thesis topic, and are chosen to illustrate the variety of possible approaches.

We have discussed the inefficiency of full order simulation methods for representing the state of physical systems, and the way pre-acquired data can be exploited to construct better representation spaces. We have shown that a range of Machine Learning methods were available for the identification of suitable reduced spaces, each with its advantages and drawbacks. Most importantly, we have underlined the differences in terms of performance and interpretability between the linear proper order decomposition method (Section 2.2.2) and neural Autoencoders (Section 3.1.5) as these methods both offer a solution to the same problem, while presenting very different properties.

The last Sections discussed the various ways dynamical models could be constructed in the absence of an already established model, either through principled approaches based on well-established theory or through direct approximation of the dynamics using Machine Learning techniques. Once again,

we underline the existence of a tradeoff between powerful approximation methods that provide good accuracy close to their training conditions, such as recurrent neural networks, and the use of theoretically grounded models that provide better guarantees but might be less accurate.

As mentioned previously, the work presented in this thesis focuses on this tradeoff as we strive to derive theoretically sound ways to combine the approximation powers of neural networks with interpretable modeling approaches, the main goal being the construction of reduced models with improved accuracy that retain theoretical guarantees. In the following Chapter, we introduce one of the main contributions of this thesis, the *CD-ROM* approach, which focuses on addressing the shortcomings of the Galerkin projection method presented above, while ensuring that the model retains as much of the properties of the original system as possible.

CHAPTER 4

CD-ROM: COMPLEMENTARY DEEP - REDUCED ORDER MODEL

Contents

4.1 Introduction 56
4.1.1 Context 56
4.1.2 Related Work 57
4.2 Model reduction approach 59
4.2.1 POD Galerkin 59
4.2.2 Non-Markovianity and Takens' theorem 60
4.3 Data driven learning of the residual 63
4.3.1 Neural Networks 63
4.3.2 Memory time scales 63
4.3.3 Training strategy 64
4.3.4 Implementation details 65
4.4 Interpretation of the model 68
4.4.1 The Mori-Zwanzig formalism 68
4.4.2 Deep Learning interpretation of the model 69
4.5 Case presentation and reduced models 71
4.5.1 Flow over a cylinder 71
4.5.2 Fluidic pinball 73
4.5.3 Parametric Kuramoto-Sivashinsky equation 75
4.6 Results and discussion 77
4.6.1 Cylinder case 78

4.6.2 Fluidic pinball results 80
4.6.3 Parametric KS equation results 82
4.7 Additional Results 86
4.7.1 Number of modes for the fluidic pinball reduction . . 86
4.7.2 Computational cost 88
4.8 Conclusion 90

4.1 Introduction

4.1.1 Context

This Chapter presents one of the main contributions of the thesis, the CD-ROM method, and is based on the work presented in Menier et al. [191]. In the previous Chapter, we discussed the importance of hybridization for the modeling of dynamical systems. We showed that linear dimensionality reduction methods could be used in combination with the POD-Galerkin method to construct reduced-order models based on the governing equations of the system at hand (see Section 3.3).

We also discussed the shortcomings of the POD-Galerkin method and showed that it is embedded with approximation errors, which limits the applicability of the method to strongly nonlinear dynamics. Thus, we introduce in this Chapter a model order reduction strategy built around the improvement of the POD-Galerkin method. The method is based on a neural closure model which only learns the *complement* of the Galerkin reduced order model, rather than the full dynamical model. The method is thus named *CD-ROM: Complemented Deep - Reduced Order Model*.

We will show in this Chapter that the CD-ROM method uses Neural Networks to correct the imperfect dynamics of Galerkin reduced order models, while retaining coherence with existing theoretical results. Moreover, the CD-ROM approach yields a time-continuous model that can be simulated as a standard dynamical system, as it is based on a time-continuous memory architecture. This should be put in contrast with classical neural sequential modeling methods such as Recurrent Neural Networks which learn a discrete time-stepping scheme for the dynamics of a system (see Section 3.2.1). Framing the model in the context of hybridization, which is at the core of this thesis, we place the model on the *hybridization spectrum* displayed in figure 4.1 and introduced in the previous Chapter to facilitate the comparison with existing modeling approaches.

Before presenting the derivation of the model, the following Section provides a direct comparison with the existing works closest to the CD-ROM method.

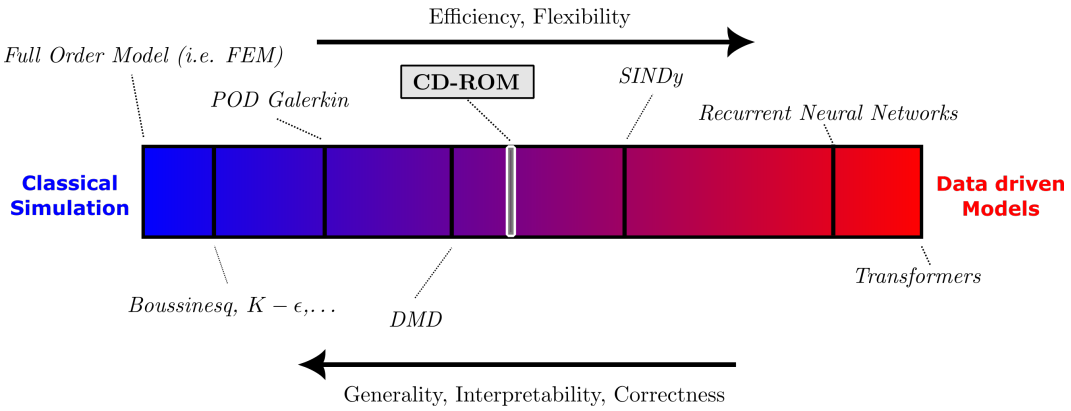


Figure 4.1: The CD-ROM method on the hybridization spectrum. *NB: This figure is provided for illustrative purposes and helps frame our proposals within the broader context of the thesis. The placement of each method on the spectrum is open to discussion, and small variations could lead to displacing a given approach closer to one end or the other.*

4.1.2 Related Work

Reduced order modeling methods offer powerful and straightforward industrial applications, thus, their improvement has been the subject of a large body of literature. We divide the existing solutions into *intrusive* and *non intrusive* methods. *Intrusive* approaches aim at learning a closure model for the available ROM such that high fidelity data are fitted, while *non intrusive* approaches are closer to those presented in Section 3.2.1 as they propose to completely replace pre-existing models with learned black-box forecasting methods. Several works have proposed to leverage deep learning methods to develop non intrusive forecasting models on low dimensional spaces and represent physical simulation problems ([184, 140, 123, 157, 131]). Specifically, the necessity of exploiting temporal information to reconstruct accurate dynamics has been underlined in [154, 164, 129, 149]. More theoretically grounded works have also proposed to develop intrusive non-Markovian closure models for existing Reduced Order Modeling approaches ([148, 143]). They are motivated by the Mori-Zwanzig formalism ([29]), which provides a theoretical framework for the modeling of partially observed systems. Similar to these efforts, our work aims at developing a theoretically grounded reduced modeling method for the forecasting of physical systems, preserving the physical insights provided by the projected PDE describing the problem. Most importantly, we focus on the fact that the majority of the approaches cited above propose to use recurrent neural networks to either learn a closure model or the full dynamics of the system. Although this choice can be motivated by the necessity of capturing memory effects, RNNs are limited for the modeling of dynamical systems, mainly because of their time-discrete nature (see Section 3.2.1). On the other hand, our

	Non-Markovian	Non Intrusive	Continuous	Memory interpretability
Wang *et al.* ([148])	✓			✓
Pawar *et al.* ([143])	✓			
Maulik *et al.* ([140])	✓	✓	✓	
Vlachas *et al.* ([201])	✓	✓		✓
Wu *et al.* ([149])	✓	✓		✓
Maulik *et al.* ([164])	✓	✓		
Pawar *et al.* ([123])	✓	✓		
CD-ROM	✓		✓	✓

Table 4.1: Comparison of closure approaches from the recent literature.

proposal is constructed around a time-continuous memory formulation with numerous advantages over discrete time models. Moreover, we underline below the higher degree of interpretability of our solution. The main novelties of the method presented in this Chapter are listed below and Table 4.1 also provides a summary of the potential crossovers between existing methods and the present work.

- **Intrusivity**: Closure modeling has been a topic of interest since the early days of numerical simulation, thus, developing *intrusive* correction models which combine with existing physical models is not a novel approach in and of itself. However, most deep learning approaches do not take this route and propose to learn forecasting models from scratch, ignoring the underlying physical laws. In this work, we show that the Galerkin ROM can be used inside the training loop to optimize the closure model in an *a-posteriori* fashion, that is, by simulating the whole model and assessing its performance. Embedding the existing ROM in the training strategy allows us to leverage pre-existing physical information rather than replace it with a physics-agnostic model.

- **Continuity**: The proposed model is embedded with a novel time continuous memory formulation. This increases the applicability of the model as it can be plugged in any initial value problem solver without being biased against specific time-step and/or numerical scheme choices made during training. Moreover, this flexibility implies that this work can be used to model arbitrarily stiff problems through the use of adaptive time-stepping schemes. Finally, the continuous structure allows the model to be used in combination with irregularly spaced data, often encountered in real-life problems, with no additional considerations to interpolation between samples.

- **Memory Interpretability**: Contrary to classical recurrent neural networks

such as the Long Short Term Memory (LSTM) used in the state of the art, the proposed memory formulation was specifically designed for numerical simulation purposes. Particularly, the time evolution of the memory has a closed form solution, which means that the memory term can be initialised to any desired degree of precision. Our proposed formulation also allows for the evaluation of the *time persistence* of information in memory, increasing the overall interpretability of the model.

- **End to End Training**: In contrast with other works which propose to learn the closure model in an *a-priori* fashion, *i.e.* by learning the dynamics correction as a standalone regression problem, we integrate the imperfect model directly within the training strategy. Thus the correction model accurately learns to compensate for the sensitivity of the Galerkin ROM and account for the long term effects of unstable, low energy modes in the system. Indeed, it has been observed in the literature that this strategy lead to more stable and accurate models ([165, 90, 147]).

The outline of the Chapter is as follows: a reminder on the POD-Galerkin reduction method is provided in Section 4.2. Section 4.3 details the main contribution of the method with the derivation of a continuous in time correction architecture for reduced order models. Motivation for our work through comparisons with existing approaches is provided in Section 4.4, while the selected test cases and results are respectively discussed in Section 4.5 and 4.6. Section 4.8 concludes the Chapter.

4.2 Model reduction approach

4.2.1 POD Galerkin

As mentioned in Section 2.2.2, the POD method can be used to construct a matrix $V_r \in \mathbb{R}^{N\times r}$ whose columns form a low dimensional basis on which the high dimensional state of a physical system $\boldsymbol{u} \in \mathbb{R}^N$ can be projected so that $\boldsymbol{a} = V_r^{\mathsf{T}}\boldsymbol{u}$ and $\boldsymbol{u} \approx \widetilde{\boldsymbol{u}} = V_r\boldsymbol{a}$. Section 3.3 also discussed the way this linear basis of principal modes could be used to project the governing equations using the Galerkin method to obtain a low dimensional system of equations for the reduced state of the system α, which in the case of the incompressible Navier Stokes equations, yields a simple algebraic model (see Section 3.3 for the derivation):

$$\frac{\mathrm{d}a_i(t)}{\mathrm{d}t} = \sum_{j=1}^{r} \widetilde{\mathcal{B}}_{i,j}\, a_i(t) + \sum_{j=1}^{r}\sum_{k=1}^{r} \widetilde{\mathcal{Q}}_{i,j,k}\, a_j(t)\, a_k(t) + \epsilon_i(t), \qquad \forall\, i = 1,\dots,r. \tag{4.1}$$

Where $\widetilde{\mathcal{B}} \in \mathbb{R}^{r\times r}$ and $\widetilde{Q} \in \mathbb{R}^{r\times r\times r}$ respectively correspond to the reduced dissipative and advective terms of the full-order equations. Unfortunately, models

constructed with the Galerkin method were shown to be embedded with error, denoted as ϵ in the equation above. In Section 3.3, we showed that this error could be directly related to the complement of the projection on the POD basis ($\boldsymbol{u} - \widetilde{\boldsymbol{u}}$), which is unavailable during the simulation of a reduced order model, as it is orthogonal by construction to the resolved representation space. In native POD-Galerkin reduced models, this error term is generally ignored. Unfortunately, this approximation means that small errors on the dynamics will compound over time and lead to significant discrepancies between the true and simulated trajectories. This is especially true in the case of nonlinear dynamical systems where orthogonal projection on the POD basis can suppress an important part of the dynamics. In the case of the Navier Stokes equations, POD-Galerkin models have been shown to fail to reproduce the dynamics even in simple cases like the flow over a cylinder [32].

Thus, the aim of this work is to retain the simplicity of the POD-Galerkin method, and learn the **complement** of the **ROM** using **deep** learning methods, we then call our proposed method **Complemented Deep - Reduced Order Model**.

4.2.2 Non-Markovianity and Takens' theorem

The residual depends on information from a subspace orthogonal to the span of the POD basis. This means that an accurate correction model cannot be directly computed from the reduced state $\boldsymbol{a}(t)$. However, we leverage the fact that the information lost by projection of the full order state can be retrieved by considering past states of the system. This hypothesis is formalized by the Takens' theorem ([12]), which states that, under mild conditions, the dynamics of a state vector can be reproduced by constructing a time-embedding from time-lagged observables: $\mathbf{z}(t) = \Big(z(t), z(t-\tau), z(t-2\tau), \dots, z(t-k\tau)\Big)$, with k large enough.[1] This is illustrated in Figure 4.2, which shows the Lorenz attractor observed via its embedded X-component. By constructing a 3-dimensional embedding of the obtained time-series, an attractor is obtained, which preserves the topology of the true attractor (*e.g.*, symmetries, correlation dimension, etc.).

This suggests that the complement of POD-Galerkin reduced order models, which we denote as $\mathcal{R}$, is non-Markovian and should consider past states of the system $\Big(\boldsymbol{a}(t-\tau), \boldsymbol{a}(t-2\tau), \dots\Big)$. However, such a discrete time-embedding is not well suited in the context of continuous time models such as equation

[1] We tacitly assumed here that z is a suitable observable. Observability analysis goes beyond the scope of this work. Nevertheless, it is well known that higher harmonic POD modes are enslaved to dominant POD modes ([105, 155]). Therefore, in the rest of the Chapter, we will assume that the dynamics of the unobserved space spanned can be retrieved from past observations of the dynamics in the space defined by the POD modes.

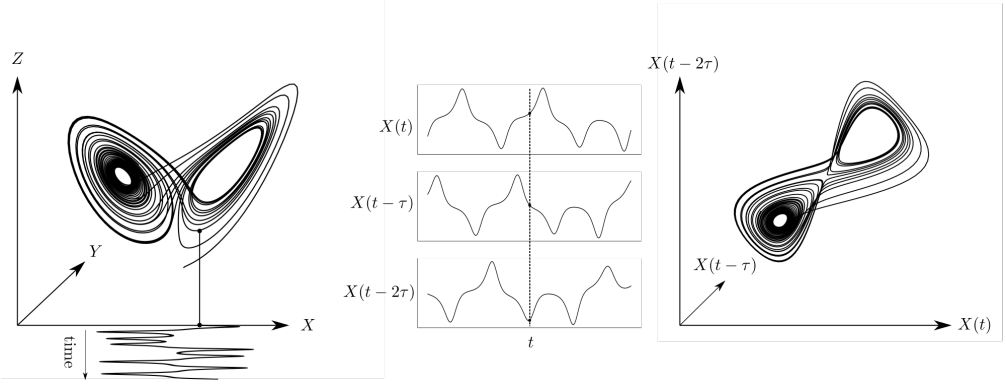


Figure 4.2: Illustration of the Takens' delay embedding theorem. Left: Original Lorenz attractor; middle: Delayed time series of the X coordinate; right: Reconstructed attractor.

(4.1). Indeed, using discrete time steps in combination with adaptive step time marching schemes, such as the Runge-Kutta method [11], would require specific considerations about interpolation between the simulation steps and the required embedding steps. To address this time-continuity issue while retaining a non-Markovian correction structure, we propose to use delay differential equations (DDE) with a continuous embedding of the past information:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a}(t) = \boldsymbol{g}(\boldsymbol{a}(t), \mathbf{y}(t)) \quad , \quad \mathbf{y}(t) = \int_{-\infty}^{t} e^{(\tau - t)\lambda}\boldsymbol{a}(\tau)\mathrm{d}\tau \tag{4.2}$$

with $\lambda \in \mathbb{R}_+$ sufficiently large for the integral above to be bounded.

These equations retain information from past states of the system in a time-continuous manner and are used for a number of modeling applications such as epidemiology or population dynamics ([18, 25]). In this formulation, the dynamics $\boldsymbol{g}$ depend both on the partially observed states of the system $\boldsymbol{a}(t)$ and a memory variable $\mathbf{y}(t)$, corresponding to the integral of past observables, damped in time by an exponential decay. In fact, the memory term $\mathbf{y}$ can be defined in many ways, depending on the problem at hand. However, the exponential decay formulation was chosen because *i*) it provides the model with the ability to consider recent states of the system while older observations are discarded and *ii*) it can be solved by directly augmenting the original system with a second ODE for the memory:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a}(t) &= \boldsymbol{g}(\boldsymbol{a}(t), \mathbf{y}(t)), \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y}(t) &= \boldsymbol{a}(t) - \lambda\mathbf{y}(t). \end{aligned} \tag{4.3}$$

The exponential decay $e^{(\tau - t)\lambda}$ acts as a filter of width $1/\lambda$ on the observables evolution and we show in a later Section that the value of the decay rate (λ) can

be learned in a data driven setting. Other applications of the same augmented ODE exist in literature but with different purposes such as modelling of subgrid-scales in Large-Eddy Simulations (LES) [33] or to find unstable-steady solutions of the Navier-Stokes equations [36]. As mentionned earlier, we have made the convenient choice of the exponential kernel since it can be described by a linear ODE and easily constrained to model dissipative dynamics, but alternative kernels exist in the DDE literature which could be considered if the exponential kernel became too constraining for certain cases.

It is worth noting that the memory variables $\mathbf{y}$ have the same dimension as the observations $\boldsymbol{a}$ in (4.3). This limitation might introduce a significant information bottleneck in the model. Indeed, Takens' theorem states that the dimension required to obtain a satisfactory embedding can go as high as twice the intrinsic dimension of the true attractor. Although there is no similar result for the continuous case, the limited dimension of the memory $\mathbf{y}$ may prevent deriving an accurate correction model. As a result, we define an encoding map $\mathbf{E} : \mathbb{R}^r \to \mathbb{R}^{n_E}$, used to lift the observations $\boldsymbol{a}$ to a higher dimensional space to increase the dimension of the memory:

$$\mathbf{y}(t) = \int_{-\infty}^{t} e^{(\tau - t)\lambda} \mathbf{E}(\boldsymbol{a}(\tau)) \mathrm{d}\tau. \tag{4.4}$$

In fact, the use of such an encoding map to unfold non-linear dynamics and recover a linear ODE is rooted in Koopman theory as each encoded coordinate can be considered as an observable of the original state. While approaches such as dictionnary learning ([77]) and kernel methods ([72]) have been proposed to learn these observables, we use neural networks to avoid additional optimization considerations and retain flexibility, similar to the works of [88, 142]. Using the modified DDE architecture (Eq.(4.4)) to close reduced order models, the correction operator $\mathcal{R}$ acting on the memory $\mathbf{y}(t)$ becomes an application from memory space to phase space: $\mathbb{R}^{n_E} \to \mathbb{R}^r$. Finally, the following augmented reduced order model is obtained:

$$\begin{array}{rcccl} \dfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a}(t) & = & \mathbf{V}_r^{\mathsf{T}} \boldsymbol{F}(\mathbf{V}_r \boldsymbol{a}) & + & \mathcal{R}(\mathbf{y}), \\ \dfrac{\mathrm{d}}{\mathrm{d}t}\mathbf{y}(t) & = & \mathbf{E}(\boldsymbol{a}) & - & \lambda \mathbf{y}. \end{array} \tag{4.5}$$

This proposed augmented ROM architecture has a similar form to the Mori-Zwanzig formalism [29] which derives an equation for the dynamics of partially observed systems. Although the parallel with the CD-ROM approach is not formally established, we discuss these similarities in Section 4.4.1 as it provides additional motivation for the above choices. Finally, the augmented ROM formulation is summarised in Figure 4.3 to help illustrate the idea.

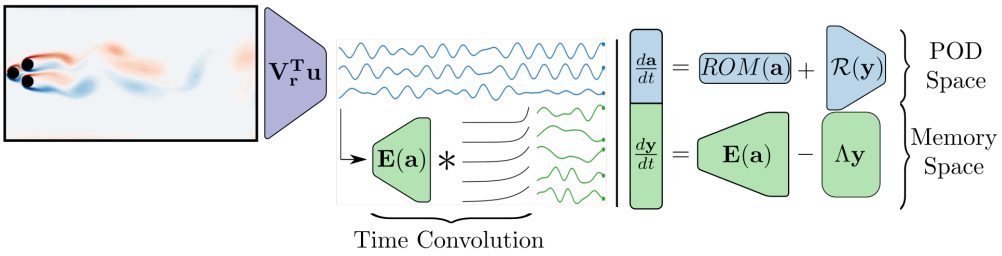


Figure 4.3: Visualisation of the CD-ROM approach. The full order solution snapshots are projected on the POD basis to obtain time series of the reduced coordinates ($\boldsymbol{a}$). These reduced coordinates are then augmented with a memory $\mathbf{y}(t)$. Finally, the augmented dynamical model is presented on the right hand side of the image.

4.3 Data driven learning of the residual

Although the motivation for the existence of the residual $\mathcal{R}$ and encoding map $\mathbf{E}$ was outlined above, little information about their form can be derived from the previous expressions. In this context, we leverage the universal function approximator property of artificial neural networks [20] to model the missing terms.

4.3.1 Neural Networks

Our work relies on the Multi Layer Perceptron architecture introduced in Section 3.1. This very simple network architecture can be used to learn any smooth continuous application $\boldsymbol{\Phi} : \mathbb{R}^r \to \mathbb{R}^k$ by optimizing the weights of a sequence of L layers:

$$\boldsymbol{\Phi}(\boldsymbol{a}) = \phi_L \circ \phi_{L-1} \circ \cdots \circ \phi_1(\boldsymbol{a}) \tag{4.6}$$

$$\phi_l = \sigma_l(\boldsymbol{W}_l \phi_{l-1} + \boldsymbol{B}_l), \qquad \forall\, l \in \{2, \ldots, L\} \tag{4.7}$$

where σ_l is a nonlinear activation function, and the dimension of ϕ_l corresponds to the number of neurons in the layer l. It has been shown that, provided the dimension of the layer is high enough, the trainable parameters $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ can be optimized to approximate any function [20]. The encoder $\mathbf{E}(\boldsymbol{a})$ and the residual $\mathcal{R}(\mathbf{y})$ are both approximated with neural networks with parameters $\boldsymbol{\theta}_\mathbf{E} = \left(\{\boldsymbol{W}_l, \boldsymbol{b}_l\},\ l = 1, .., L_\mathbf{E}\right)$ and $\boldsymbol{\theta}_\mathcal{R}$ respectively.

4.3.2 Memory time scales

Physical systems often involve a variety of phenomena each evolving at different time scales. Capturing these phenomena can be critical to accurately model

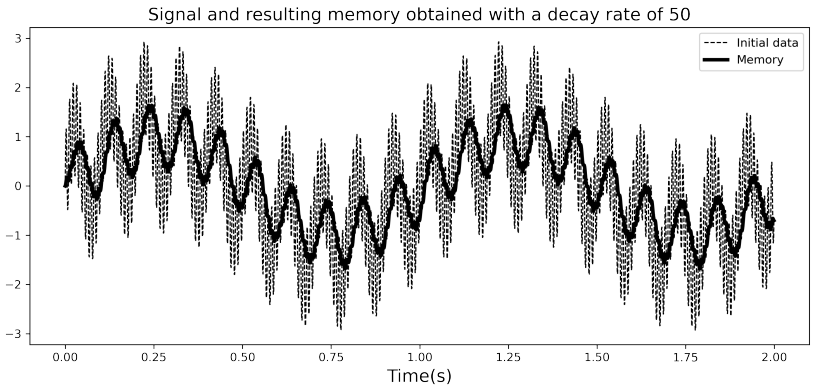


Figure 4.4: Superposition of sinusoidal signals of frequencies 1, 10 and 100 Hz, filtered by an exponential decay with a decay rate $\lambda = 50$. The higher frequency (100 Hz) is filtered out while the other two are reproduced in the memory signal. This implies that recent events in memory due to the 100 Hz frequency are filtered out so that the memory is mostly driven by *older* events associated with lower frequencies. Note that the memory signal has been scaled by a factor λ for clarity.

these systems, which is why the memory should be able to retain information at different rates. The time scales accounted for by the memory are driven by the parameter λ of the exponential decay in Eq. (4.2), which acts as a low-pass filter on the encoded trajectory (see Figure 4.4 for an illustration).

To retain information at different rates, λ can be adjusted for each observable $E_i(\boldsymbol{a})$. Equation (4.5) is then modified accordingly with the single λ parameter replaced with a diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}_+^{n_E \times n_E}$ whose entries can be trained to select time scales relevant to the system at hand. This finally results in the CD-ROM architecture:

$$\begin{array}{rcccl} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a}(t) & = & \mathbf{V}_r^{\mathsf{T}}\boldsymbol{F}(\mathbf{V_r}\boldsymbol{a}) & + & \mathcal{R}(\mathbf{y};\boldsymbol{\theta}_{\mathcal{R}}), \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y}(t) & = & \mathbf{E}(\boldsymbol{a};\boldsymbol{\theta}_{\mathbf{E}}) & - & \boldsymbol{\Lambda}\,\mathbf{y}. \end{array} \tag{4.8}$$

4.3.3 Training strategy

Optimizing the parameters of the model (4.8) requires consideration of past states of the system as well as their impact on the dynamics in the future. In some sense, the problem is similar to the optimization of classical recurrent neural networks for the simulation of dynamical systems [201]. The major difference is that the aim of the present work is to derive a continuous time dynamical model, while recurrent networks have traditionally been used to model transitions between discrete time instants. As a result, we cannot use standard backpropagation through time to optimize the model. Instead, the NeuralODE

approach [99] introduced in detail in Section 3.2.3, is used.

As explained in Section 4.2.1, reduced models of the incompressible Navier-Stokes equations can be expressed directly in terms of the reduced coordinates $\boldsymbol{a}$ and evaluated through simple tensorial expressions. This means that the ROM dynamics can be directly computed and back-propagated through once the reduced operators $\widetilde{\mathcal{B}}$ and $\widetilde{\mathcal{Q}}$ are assembled (see Eq.(4.1)). The Deep ROM architecture (Eq.(4.8)) can then be optimized within the Neural ODE framework by concatenating the reduced state and the memory into a single state vector $\mathbf{z}(t) = [\boldsymbol{a}(t), \mathbf{y}(t)]$, with dynamics $\boldsymbol{f}(\mathbf{z}; \boldsymbol{\theta}_{\mathbf{E}}, \boldsymbol{\theta}_{\mathcal{R}}, \boldsymbol{\Lambda})$:

$$\boldsymbol{f}(\mathbf{z}; \boldsymbol{\theta}_{\mathbf{E}}, \boldsymbol{\theta}_{\mathcal{R}}, \boldsymbol{\Lambda}) = \begin{pmatrix} \widetilde{\mathcal{B}}\boldsymbol{a} + \boldsymbol{a}^{\mathsf{T}}\widetilde{\mathcal{Q}}\boldsymbol{a} & + & \mathcal{R}(\mathbf{y}; \boldsymbol{\theta}_{\mathcal{R}}) \\ \mathbf{E}(\boldsymbol{a}; \boldsymbol{\theta}_{\mathbf{E}}) & - & \boldsymbol{\Lambda}\mathbf{y} \end{pmatrix}. \tag{4.9}$$

Since the last modes in the POD basis $\mathbf{V}$ are typically associated to low-energy dissipative scales [151], if r is not large enough, the ROM model retrieved by $\mathbf{V}_r$ might be unstable and diverge after a few time integration steps. This potentially unstable primal model is however embedded in the optimization framework so that the residual is trained accordingly, resulting in a stable, accurate, model.

4.3.4 Implementation details

This Section details the various practical choices made to implement and train the CD-ROM model.

Training data

To optimize the parameters of equation (4.9) through the Neural ODE approach, one only needs knowledge of the true trajectory of the reduced coordinates $\boldsymbol{a}^\star(t)$. As presented in Section 2.2.2, this trajectory data can be obtained by projecting the solutions computed with the full order solver on the POD basis:

$$\boldsymbol{a}^\star(t) = \mathbf{V}_r^{\mathsf{T}}\mathbf{u}(t). \tag{4.10}$$

If the snapshots are sampled with a time interval Δ_t, simulating the corrected ROM (4.9) for n_t time steps leads to the following L^2 mean squared error:

$$J = \frac{1}{n_t}\sum_{i=1}^{n_t} \|\boldsymbol{a}(i\,\Delta_t) - \boldsymbol{a}^\star(i\,\Delta_t)\|_2^2 \tag{4.11}$$

for which a gradient can be computed by integrating the adjoint equation (3.47).

Memory Initialisation

For the model to be accurate, the memory term needs to be initialized by evaluating the memory integral at time $t = 0$ which requires knowledge of the past of the true trajectory:

$$\mathbf{y}(0) = \int_{-\infty}^{0} \mathbf{E}(\boldsymbol{a}^{\star}(\tau); \theta_{\mathbf{E}}) e^{\mathbf{\Lambda}\tau} \mathrm{d}\tau. \tag{4.12}$$

To be able to compute this integral, the infinite horizon of the memory integral can be relaxed by defining a finite $\tau_{\min}$ from the longest time scale $\lambda_{\min}$ of the matrix $\mathbf{\Lambda}$ and a threshold $\epsilon \ll 1$, chosen to be small enough such that the relative error made on the initial memory is sufficiently small:

$$\tau_{\min} = -\frac{\log \epsilon}{\lambda_{\min}}, \tag{4.13}$$

$$\mathbf{y}(0) \approx \int_{-\tau_{\min}}^{0} \mathbf{E}(\boldsymbol{a}^{\star}(\tau); \theta_{\mathbf{E}}) e^{\mathbf{\Lambda}\tau} \mathrm{d}\tau. \tag{4.14}$$

Because this initial memory term directly depends on the parameters of the encoder and the matrix $\mathbf{\Lambda}$, it needs to be re-computed at each training epoch. This can be done very efficiently on a GPU through a simple trapezoidal approximation of the integral in Eq. (4.14). It should be noted that, while the initialization of the memory is necessary to obtain an exact model of the system at hand, it could limit the applicability of the method to certain real life settings. However, there are ways to make it less critical, such as initializing the memory with white noise during training. This would of course impact the accuracy of the model, depending on the system at hand.

Training Loop

Algorithm 1 summarises the NeuralODE training procedure.

Residual regularization

We observed that only training the model to follow the trajectory data can lead to poor local optima with large corrections applied to the original model. Since the magnitude of the correction is meant to be small when r is sufficiently large, this tends to indicate over-fitting. This is an issue as such a model does not capture the true dynamics, and will quickly diverge when evaluated on conditions different from the training trajectory. To address this, a regularization term can be added to the loss, to limit the magnitude of the corrections applied to the ROM:

Algorithm 1 Training the CD-ROM as a Neural ODE

Require: $f(\mathbf{z};\theta)$ the dynamics of the CD-ROM, $\mathcal{L}$ the loss, η the learning rate

for $i \leftarrow 1$ to N **do** ▷ Training iterations

Sample a batch of trajectories $\boldsymbol{a}^\star_{-\tau_{min}\rightarrow T}$

Compute the initial memory $\mathbf{y_0}$ ▷ Equation 4.14

$\mathbf{z_0} \leftarrow [\boldsymbol{a}^\star_0, \mathbf{y_0}]$

$[\boldsymbol{a}_t, \mathbf{y}_t] \leftarrow \mathbf{z_0} + \int_0^t f(\mathbf{z};\theta)\mathrm{d}t$ ▷ CD-ROM simulation

Compute the loss $\mathcal{L}(\boldsymbol{a}_{0\rightarrow T}, \boldsymbol{a}^\star_{0\rightarrow T})$

$\mu_T \leftarrow \frac{d\mathcal{L}}{d\mathbf{z}_T}$ ▷ Adjoint Initial Condition

$\mu_t \leftarrow \mu_T - \int_T^t \mu^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{z}_t}\mathrm{d}t$ ▷ Adjoint Equation

$\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L} \leftarrow \int_0^T -\mu_t^\mathsf{T} \frac{\partial \boldsymbol{f}}{\partial \theta}\mathrm{d}t$ ▷ Compute the Gradient

$\theta \leftarrow \theta - \eta \frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}$

end for

$$J = \frac{1}{n_t}\sum_{i=1}^{n_t}\Big(\|\boldsymbol{a}(i\,\Delta_t) - \boldsymbol{a}^\star(i\,\Delta_t)\|_2^2 + \rho\,\|\mathcal{R}(\mathbf{y}(i\,\Delta_t)\|_2^2\Big) \tag{4.15}$$

where n_t is the number of time steps of length Δ_t in the optimized trajectory, $\boldsymbol{a}^\star$ is the training trajectory data, and ρ is a hyper-parameter chosen to balance the importance of the regularization *w.r.t.* the rest of the loss. Although this fairly simple approach already helped guide the training, we obtained better results by adding a custom regularization loss. This loss is based on the approximate value of the residual on the true trajectory, which can be obtained by computing the time-derivative of the true reduced coordinates $\dot{\boldsymbol{a}}^\star$ through finite differences, and computing the difference with the derivative defined by the uncorrected ROM:

$$\mathcal{R}^\star = \dot{\boldsymbol{a}}^\star - \widetilde{\mathcal{L}}\boldsymbol{a}^\star - \boldsymbol{a}^{\star\mathsf{T}}\widetilde{\mathcal{Q}}\boldsymbol{a}^\star. \tag{4.16}$$

In parallel, the true value of the memory can be evaluated from the true trajectory integrating in time the following ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y} = \mathbf{E}(\boldsymbol{a}^\star;\boldsymbol{\theta}_\mathbf{E}) - \boldsymbol{\Lambda}\mathbf{y}. \tag{4.17}$$

One can then define a regularization term $\mathcal{L}_{\mathrm{corr}}$ for the loss:

$$\mathcal{L}_{\mathrm{corr}} = \frac{1}{n_t}\sum_{i=1}^{n_t}\|\mathcal{R}(\mathbf{y}(i\Delta_t)) - \mathcal{R}^\star(i\Delta_t)\|_2^2\,. \tag{4.18}$$

This regularized loss definition was observed to lead to models with better generalization properties. Note that $\mathcal{L}_{\mathrm{corr}}$ can be computed for a small random

subset of the training batch at each epoch, to accelerate training while keeping a "*stochastic*" regularization for the residual.

Encoded space regularization

Regularizing the encoded space can both help smooth the training process and increase the robustness of the model to unseen conditions. Taking inspiration from existing work ([88]), we propose to add the identity function to the encoder model. This means that useful information is embedded in memory in the form of a time convolution of the past resolved states of the system. The encoded state $\mathbf{E}(\boldsymbol{a}_t)$ is then composed of both the reduced state $\boldsymbol{a}_t$ and a learnable nonlinear transformation $\mathcal{MLP}(\cdot, \boldsymbol{\theta}_{\mathbf{E}})$ of it:

$$\mathbf{E}(\boldsymbol{a}_t, \theta_{\mathbf{E}}) = \left[\boldsymbol{a}_t, \mathcal{MLP}(\boldsymbol{a}_t, \boldsymbol{\theta}_{\mathbf{E}})\right] \tag{4.19}$$

With this structure, the encoder will only be learning additional nonlinear transformations of the state, simplifying the training and introducing a level of structure in the encoded space as we ensure its first dimensions are coherent with the phase space.

Memory Dimension

The dimension of the memory in the CD-ROM formulation is a hyperparameter that should be chosen depending on the case and the dimension of the reduced state. Choosing an excessively low memory dimension will lead to poor prediction performance, while using too high of a dimension will negatively impact the computational cost of the corrected model and might lead to overfitting and poor numerical conditioning.

In our experiements, good results were obtained using memory dimensions ranging from $2\times$ to $10\times$ the dimension of the reduced state.

4.4 Interpretation of the model

Before presenting the selected test cases and results, further justification and insights into the model are discussed in this Section. First, the Mori-Zwanzig formalism is introduced to frame our model in the context of dynamical systems theory. Then, we study how it can be compared to purely data driven approaches, such as reservoir computing.

4.4.1 The Mori-Zwanzig formalism

The Mori-Zwanzig formalism ([6, 10, 29]) provides a closed form for the dynamics of partially observed systems by distinguishing three separate terms:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a}(t) = \Omega(\boldsymbol{a}(t)) + \int_0^t K(\boldsymbol{a}(s))\mathrm{d}s + F(t) \tag{4.20}$$

where $\boldsymbol{a}(t)$ are observables of a system defined as a projection of the full order state onto the observable space and Ω is the projected part of the original dynamics. These two quantities can respectively be identified as the reduced coordinates and dynamics of classical POD-Galerkin models. The remaining terms account for the impact of the non-observed coordinates of the system on the resolved dynamics. K represents the dynamical exchanges between resolved and unresolved dynamics during the simulation, while F accounts for the incomplete knowledge of the initial condition and system dynamics. Under the condition that the unresolved dynamics be dissipative, which is reasonable when the unobserved coordinates correspond to the small scales of a dynamical system, and that the boundary of the integral term in Eq. (4.20) be modified to $-\infty$, the last term in the Mori-Zwanzig formalism vanishes, leading to the following formulation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a}(t) = \Omega(\boldsymbol{a}(t)) + \int_{-\infty}^t K(\boldsymbol{a}(s))\mathrm{d}s. \tag{4.21}$$

The residual model $\mathcal{R}$ proposed above can then be identified with the memory integral defined by the time-convolution kernel K, providing a strong connection with our non-Markovian correction hypothesis. Framing the CD-ROM architecture in the context of the Mori-Zwanzig formalism further justifies our modeling choices. Yet, it does not provide additional insights into the form of the correction model since little information is known about the convolution operator which can be infinite dimensional in certain cases. Choices about the structure of K need to be made. In this work, assumptions are made about the vanishing impact of past states of the system on the residual model, accounted for by the matrix $\boldsymbol{\Lambda}$ in our approach. In fact, this parallel with the Mori-Zwanzig formalism was further exploited in our work and led to the development of a second contribution, which we introduce in Chapter 6.

4.4.2 Deep Learning interpretation of the model

In this Paragraph, the link between the CD-ROM architecture and classical deep learning models is discussed. Numerous methods have been proposed to model sequential data, each relying on a specific mechanism to extract and retain meaningful information from the past states of the system. The most popular architectures, GRU and LSTM ([56, 23, 57]), both use a combination of *gating* mechanisms to learn long term dependencies in a sequence. Other approaches like reservoir computing rely on an underlying dynamical system forced by the sequence data to predict the required output. In fact, strong similarities can be identified between our approach and *echo state networks* (ESN [45, 128]), a

widespread reservoir computing architecture. ESNs are based on the simulation of random dynamics described by matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_R$. The matrix $\mathbf{W}_{\text{in}}$ is used to encode some data $\mathbf{x}_t$ in higher dimension, while the matrix W_R holds the weights of the reservoir used to advance the state (memory) $\mathbf{y}_t$ in time:

$$\mathbf{y}_{t+1} = \sigma(\mathbf{W}_{\text{in}}\mathbf{x}_t + \mathbf{W}_R\mathbf{y}_t). \tag{4.22}$$

For the sake of comparison, the nonlinear activation function σ is dropped, and we consider that equation (4.22) results from the Euler integration with time-step Δ_t of a continuous system describing the dynamics of the memory $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}$:

$$\mathbf{y}_{t+1} = \mathbf{W}_{\text{in}}\mathbf{x}_t + \mathbf{W}_R\mathbf{y}_t \tag{4.23}$$

$$= \mathbf{y}_t + \Delta_t\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} \tag{4.24}$$

which leads to the following expression:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \frac{\mathbf{W}_{\text{in}}}{\Delta_t}\mathbf{x}_t + \frac{(\mathbf{W}_R - I)}{\Delta_t}\mathbf{y}_t. \tag{4.25}$$

Because the spectral radius of the matrix W_R is constrained to be less than unity, all the eigenvalues of the operator $\overline{\mathbf{W}}_R = \frac{(\mathbf{W}_R - \mathbf{I})}{\Delta_t}$ have negative real parts, leading the ESN to have memory dissipation properties similar to those of our model. To underline this similarity, we can express the state of the ESN at time t by diagonalising the operator $\overline{\mathbf{W}}_R = \mathbf{P}\mathbf{\Lambda}_R\mathbf{P}^{-1}$:

$$\mathbf{y}(t) = \int_0^T \mathbf{P}e^{\mathbf{\Lambda}_R(t-\tau)}\mathbf{P}^{-1}\frac{\mathbf{W}_{\text{in}}}{\Delta_t}\mathbf{x}(\tau)\mathrm{d}\tau. \tag{4.26}$$

Thus, a parallel between our model and the ESN is outlined. Major differences remain in the fact that the dynamics ($\mathbf{W}_R$) and encoding matrix ($\mathbf{W}_{\text{in}}$) are not optimised during the ESN training. It should also be noted that nonlinearity is introduced in the ESN dynamics through the σ activation function, while our model is based on linear memory dynamics and on a nonlinear encoding operator. These comparisons between direct deep learning methods and the continuous correction approach help build intuition about the role of each term in the model. The encoder can be compared with the input gate of an LSTM, or the $\mathbf{W}_{\text{in}}$ matrix of the ESN, the $\mathbf{\Lambda}$ matrix provides a tunable *forget* mechanism, while the residual term $\mathcal{R}$ plays the role of the output operator in our "continuous recurrent network".

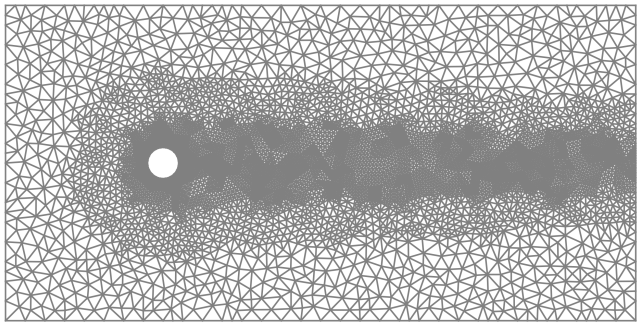


Figure 4.5: Computational domain for the cylinder case.

4.5 Case presentation and reduced models

In this Section, we introduce the simulation cases selected to demonstrate the ability of the CD-ROM approach to improve the performance of POD-Galerkin models. We first present two flow problems to illustrate the benefits of the CD-ROM architecture in the context of fluid mechanics. The first case is the standard configuration of the flow over a cylinder, often used as a benchmark for reduction methods. The second case is the fluidic pinball flow, introduced in Noack and Morzyński [89] for the development of new control strategies. Finally, we introduce the case of the 1D Kuramoto-Sivashinsky which we use to demonstrate the ability of the CD-ROM approach to extend to parametric simulation problems.

4.5.1 Flow over a cylinder

The two-dimensional incompressible flow over a cylinder has been extensively studied in the context of reduced order modeling [17, 32] and model identification [79, 105] which makes this test case a good initial benchmark for the proposed correction method. The flow is simulated at a Reynolds number of $Re = 100$ based on the cylinder diameter and the velocity of the incoming flow. In this regime, the flow is laminar and exhibits vortex shedding in the wake of the cylinder.

The flow is governed by the incompressible Navier-Stokes equations, here solved using the FEniCs finite elements solver [50, 67]. The retained mesh is shown in Figure 4.5. It is rectangle-shaped, spanning from $x = -5$ to $x = 15$ in the streamwise direction, and from $y = -5$ to $y = 5$ in the transverse direction. The cylinder has a diameter $D = 1$, centered around the origin. The inflow is modeled as a uniform axial flow ($\mathbf{u}(-5, y) = [U_\infty, 0]$) and a free-slip condition is used for the lower and upper boundaries of the rectangle while a no-slip condition is enforced at the cylinder surface. Finally, a stress-free condition is used for the outlet.

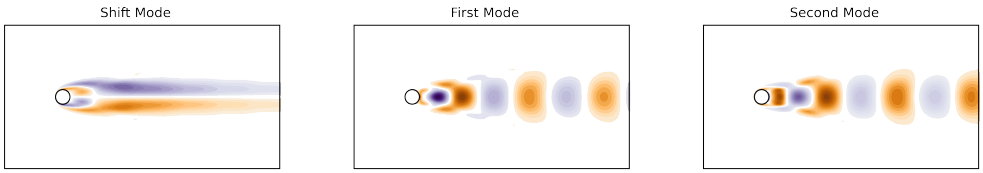


Figure 4.6: Vorticity fields of the modes selected for the reduced order modeling of the cylinder flow.

A reduced order modeling strategy is employed to obtain a baseline model for the correction approach. The main results are summarised here but we refer to Noack et al. [32] for details on the reduction strategy. The vortex shedding regime of the cylinder flow is simulated with the FEniCs solver to obtain snapshot data and compute the POD modes. The first two modes, accounting for more than 95% of the Frobenius norm of the snapshot matrix are selected. The steady solution of the system is computed with a Newton method to construct a so-called shift mode ($\boldsymbol{a}_\triangle$). This mode is computed as a vector orthogonal to the plane described by the first two modes, pointing to the base flow solution $\mathbf{u}_b$, and serves as a *support* for the simulation of the transition of the system from its steady state to the vortex shedding limit cycle. A three-dimensional POD basis is thus finally obtained. They are shown in Fig. 4.6 in terms of the vorticity field.

The training data corresponds to the simulation of the transition of the system to the limit cycle of oscillations, starting from an initial condition $\mathbf{u}_0$. This initial condition is taken as a point close to the base flow $\mathbf{u}_b$ which is the fixed point of the incompressible Navier Stokes equations. Following the procedure of Loiseau *et al.* ([105]), we chose

$$\mathbf{u}_0 = \mathbf{u}_b + \epsilon \mathbf{v}_1, \tag{4.27}$$

where $\mathbf{v}_1$ is the first POD mode, and $\epsilon > 0$ is a small coefficient used to perturb the unstable base flow.

Through Galerkin projection of the discretised Navier-Stokes equations, a system of 3 coupled ODEs is obtained, describing the dynamics of the reduced coordinates vector. The results of the simulation of the transition using both the Finite Elements model and the Galerkin ROM are displayed in Figure 4.7. Even though the three equation model is able to simulate the transient dynamics, its trajectory strongly diverges from the projected snapshot data. The transition starts much later than in the full order simulation, due to a growth rate of the ROM's transition lower than what it should be. Another significant issue with the model is its stabilization around the limit cycle ($\boldsymbol{a}_\triangle = \mathbf{0}$) where an *overshoot* can be observed before the vortex shedding regime is established, which is not observed in the snapshot data. These discrepancies between the two trajectories can be attributed to the ignored residual term in the dynamics, making this

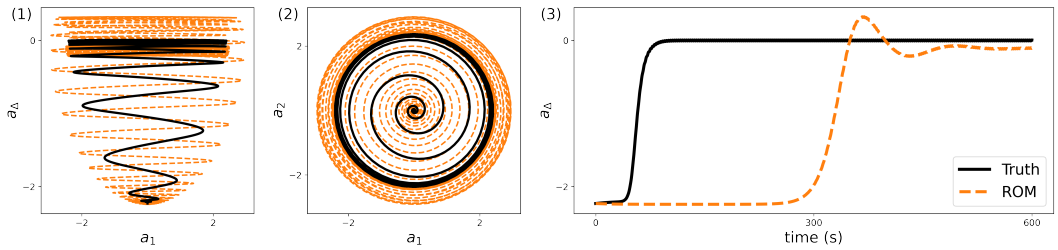


Figure 4.7: True trajectory and simulation of an uncorrected Galerkin ROM. Plots (1) & (2) show the phase space trajectories projected on the $\boldsymbol{a}_1 - \boldsymbol{a}_\Delta$ and $\boldsymbol{a}_1 - \boldsymbol{a}_2$ planes, while (3) shows the time evolution of the shift mode coefficient.

model a good baseline for the correction approach illustrated in Section 4.6.

A test trajectory is also simulated in FEniCs by taking a random initial condition in the phase space spanned by the three selected modes, such that:

$$\mathbf{u}_{0,test} = \sum_{i=0}^{3} \mathbf{v_i} a_i, \qquad a_i \sim \mathcal{N}(0,1), \tag{4.28}$$

where $\mathbf{v}_i$ are the POD modes, and a_i are random reduced coordinates sampled from a normal distribution. Starting from this initial condition, the finite element model is simulated in FEniCs for 700 seconds to ensure the system reaches the oscillation regime. The performance of the various models on this data trajectory are presented in Section 4.6.

4.5.2 Fluidic pinball

The second case used to demonstrate the approach is the so-called fluidic pinball. Initially proposed as a challenging test bed for the development of control laws [89], the fluidic pinball case offers a good trade-off between complexity of its dynamics and interpretability [116, 156]. The simulation domain (Figure 4.8) is composed of three equidistant cylinders, each generating vortices in the wake which interact to create rich dynamics. The mesh used for the simulation was provided by the authors of Cornejo Maceda et al. [100]. Displayed in Figure 4.8, the domain is a rectangle spanning from $x = -6$ to $x = 20$ in the streamwise direction, and $y = -6$ to $y = 6$ in the transverse direction. Three identical cylinders with diameter $D = 1$ are arranged in an equilateral triangle, with centers' coordinates $(0, 0.75), (0, -0.75)$ and $(-1.25, 0)$ respectively. The boundary conditions are identical to those of the cylinder case in Sec. 4.5.1. The inflow is modeled as a uniform axial flow, the upper and lower boundaries of the computational domain are modeled as free-slip, while a no-slip condition is used for the walls of the three cylinders and the outlet is modeled as stress-free.

The flow is simulated at a Reynolds number of $Re = 130$. At this Reynolds number, the flow dynamics have been shown to be chaotic [156], which makes

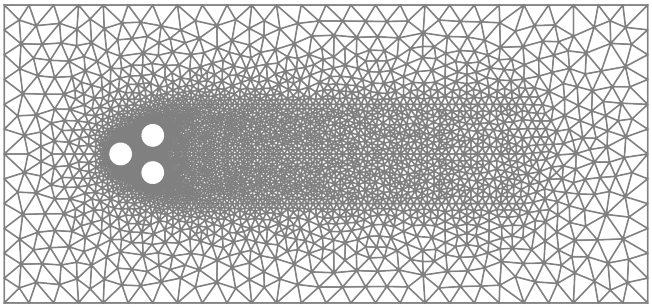


Figure 4.8: Computational domain of the fluidic pinball case.

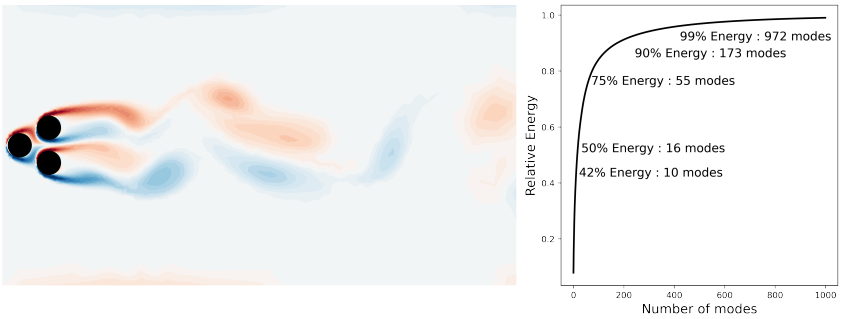


Figure 4.9: Flow complexity of the pinball case. Left: Vorticity field of the flow in the chaotic regime. Right: Spectrum of the snapshot matrix.

it a challenging problem for any correction model as the smallest error on the dynamics will make the simulated trajectory diverge exponentially fast in time from the truth. These kind of chaotic problems are starting to get traction as interesting benchmarks for forecasting tasks and modeling problems [158]. The flow is simulated for 1800 seconds in the chaotic regime, which, based on the ergodic property of the system, yields a trajectory long enough to be representative of it's attractor.

It should be noted that the pinball flow is much more complex than the cylinder, as evidenced from Figure 4.9 where it is seen that many POD modes are required to account for a significant part of the energy. Almost a thousand modes would be required to capture 99% of the Frobenius norm of the snapshot data, while only 8 are required to achieve the same accuracy in the cylinder case.

We chose to build a POD-Galerkin model of this flow using only the first 10 POD modes. Although this choice is somewhat arbitrary, it was made to challenge the correction method, as the mean reconstruction error of about 60% has an important impact on the approximated dynamics. Indeed, the obtained reduced model quickly separates from the original trajectory, as expected from a chaotic system. More problematic is the fact that the reduced model is very

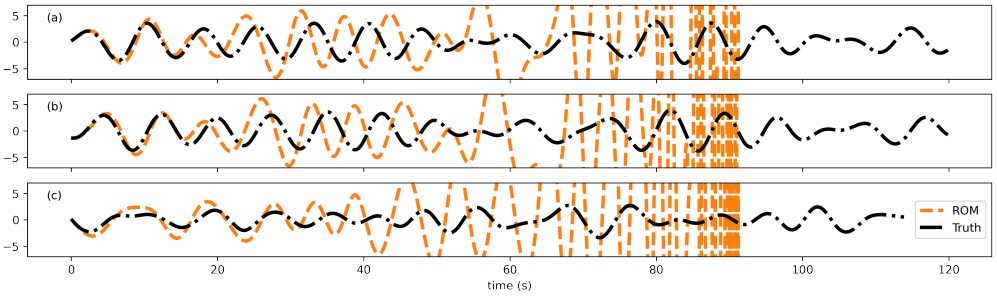


Figure 4.10: From top to bottom: time evolution of the first 3 POD coefficients. *Black line*: true value of the pinball's POD coefficients, *orange line*: trajectories obtained by integrating an uncorrected Galerkin reduced order model constructed using the first 10 POD modes.

unstable and diverges after 70 s of simulation, as shown in Figure 4.10. Application of the correction method is presented in Section 4.6, a discussion on the impact of the number of modes is provided in Section 4.7.1, while additional information on the computational cost of the approach is given in Section 4.7.2.

4.5.3 Parametric Kuramoto-Sivashinsky equation

Finally, to illustrate the ability of the proposed method to extend to parametric problems, we introduce the case of the 1D parametric Kuramoto-Sivashinsky (KS) equation. This case is often used to validate physical modeling methods as it is inexpensive to simulate because of the relatively low dimension of the discretization required, and presents non linear dynamics. Moreover, depending on the parameters used for the simulation, the dynamics become chaotic, making it significantly more challenging to forecasting approaches. The KS equation is formulated as follows:

$$\frac{\partial u}{\partial t} = -\frac{1}{2}\nabla \cdot u^2 - \Delta u - \nu \Delta^2 u, \tag{4.29}$$

$$u(x + L, t) = u(x, t), \tag{4.30}$$

$$u(x, 0) = g(x), \tag{4.31}$$

where L is the length of the 1D simulation domain, g is the initial condition and ν is a parameter that controls the degree of dissipativity of the system. Taking inspiration from Wang, Ripamonti, and Hesthaven [148], we propose to learn a corrected ROM for this problem under varying ν values. As in Wang, Ripamonti, and Hesthaven [148], we chose $L = 22$ and g is computed as the sum of the four leading Fourier modes with coefficients 0.06. The problem is discretised spatially on a basis of $N = 513$ Fourier modes, and integrated in time through the semi-implicit third order scheme from Kar [39]. This choice of discretisation

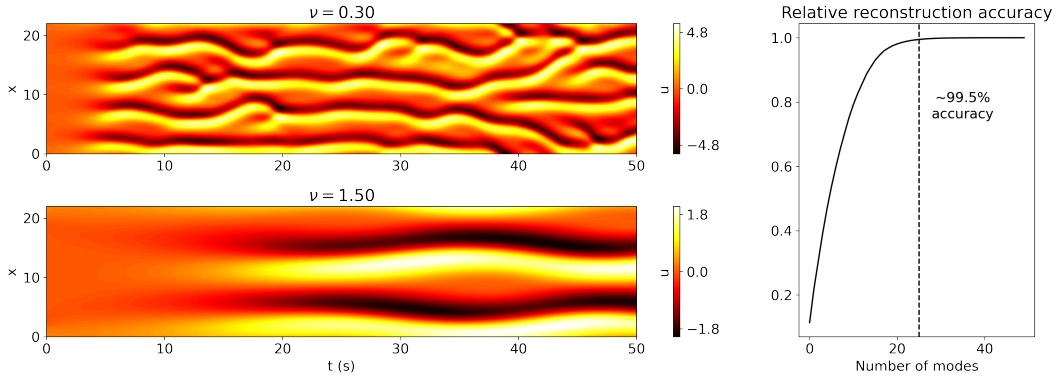


Figure 4.11: *Left*: Simulations of the KS equation carried out under two different parameter values. *Right*: Relative snapshot reconstruction accuracy against the number of POD modes used for reconstruction.

implies that the periodicity condition (equation 4.30) is satisfied by construction. The simulation is carried out for a duration of 50 seconds, using a time step (Δ_t) of $0.025s$.

As in Wang, Ripamonti, and Hesthaven [148], the parameter ν is varied in the range $[0.3, 1.5]$. As mentioned in the previous Paragraph, this parameter controls the degree of dissipation in the system, thus, low ν values lead to more chaotic dynamics and a harder model reduction task. This is represented on figure 4.11, which displays the differences between simulations carried out at the limits of the chosen parameter range.

To create a reduced model of the system, the simulation problem is solved for 25 parameter values selected within a log-linear range from $\nu = 0.3$ to $\nu = 1.5$. With this initial data, the proper orthogonal decomposition of the snapshot matrix is computed. The evolution of the relative snapshot reconstruction error depending on the number of selected modes is shown in Figure 4.11. To assemble the Galerkin reduced model, we select the 25 leading POD modes, which account for more than 99.5% of the information in the snapshot data. The computed POD modes form a basis V, which can be used to approximate the solution field $\mathbf{u}(t, \nu)$ computed for a given time and parameter value as follows: $\mathbf{u}(t, \nu) \approx \tilde{\mathbf{u}}(t, \nu) = V\boldsymbol{a}(t, \nu)$.

As in the non-parametric case, computing the vector of reduced coordinates $\boldsymbol{a}(t, \nu)$ is sufficient to fully determine the approximate solution $\tilde{\mathbf{u}}(t, \nu)$. Finally, the Galerkin projection method described in Section 4.2 is applied to the KS equation, yielding the following reduced model:

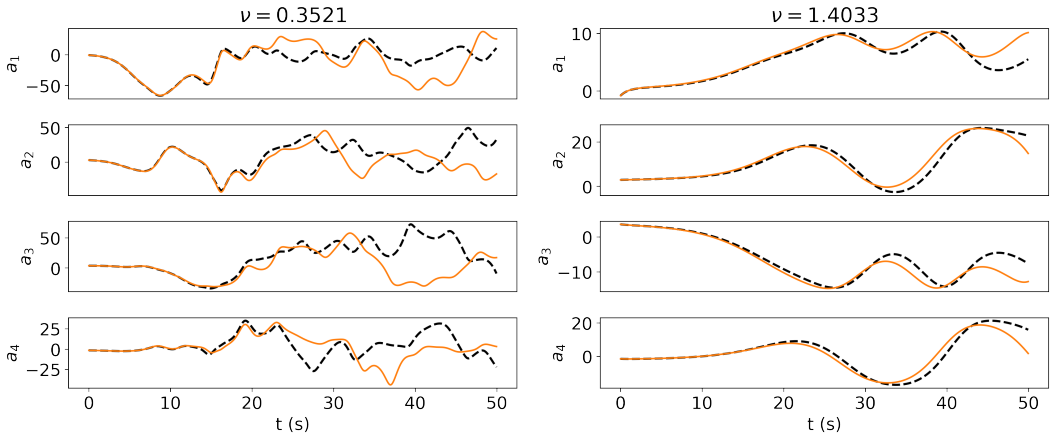


Figure 4.12: Coefficients of the first 4 POD modes simulated with the uncorrected Galerkin ROM under different parameter values. *Dashed line*: Projected DNS data, *full line*: Simulation of uncorrected the Galerkin ROM.

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = -\frac{1}{2}\boldsymbol{a}^{\mathsf{T}}\tilde{\mathcal{Q}}\boldsymbol{a} - \tilde{\mathcal{L}}\boldsymbol{a} - \nu\tilde{\mathcal{L}}^2\boldsymbol{a}, \tag{4.32}$$

$$\tilde{\mathcal{Q}}_{i,j,k} = < v_i, \nabla(v_j v_k) >, \qquad i,j,k = 1,...,r \tag{4.33}$$

$$\tilde{\mathcal{L}}_{i,j} = < v_i, \Delta v_j >, \qquad i,j = 1,...,r \tag{4.34}$$

$$\tilde{\mathcal{L}}^2_{i,j} = < v_i, \Delta^2 v_j >, \qquad i,j = 1,...,r \tag{4.35}$$

where v_i are the POD modes, and $< \cdot, \cdot >$ is an inner product defined over the computational domain. It can be noted that the $\tilde{\mathcal{Q}}$ and $\tilde{\mathcal{L}}$ operators are the one dimensional equivalent of the reduced Navier-Stokes operators introduced in Section 3.3. Similarly, the operator $\tilde{\mathcal{L}}^2$ is a linear operator corresponding to the fourth order derivative in equation 4.29. To test the model, 62 test parameter values are selected randomly in the range $\nu \in [0.3, 1.5]$ following a log-uniform distribution.

The uncorrected Galerkin ROM is simulated using the semi-implicit scheme from Kar [39]. Figure 4.12 presents the results obtained by simulating the Galerkin model under different parameter values. The figure clearly underlines the difficulty of modeling lower ν values, as we observe that the Galerkin model diverges quickly from the true trajectory. Results obtained by augmenting the Galerkin ROM with the CD-ROM architecture are presented in Section 7.

4.6 Results and discussion

In this Section, we present the results obtained by applying the CD-ROM method to the imperfect Galerkin ROMs presented in the previous Section. While some

of the design choices regarding each specific cases are discussed in the following Paragraphs, we refer the reader to Appendix 9.1 for a description of the various training details and hyper-parameter choices.

4.6.1 Cylinder case

The reduced model derived in Section 4.5 above was shown to be efficient for the simulation of the vortex shedding regime, but not suited to the simulation of transient dynamics. To apply the proposed correction procedure to this model, the modeling terms introduced in Sec. 4.3 are added to the reduced model. The dimension of the memory is chosen to be ten times the dimension of the reduced state. The residual $\mathcal{R}$ and encoder $\mathbf{E}$ are defined as multi layer perceptrons, using the Rectified Linear Unit activation function. Finally, the diagonal of the memory matrix $\mathbf{\Lambda}$ is initialised at random from a normal distribution. The model is trained with the Adam optimizer, and the trajectory loss introduced earlier (Eq. (4.11)) is used in combination with the additional regularization terms introduced in Section 4.3.4.

As described in Section 4.5, the model is trained on the trajectory data obtained by simulating the transition from an initial condition close to the base flow. Since this target trajectory starts close to the base flow of the system, which is stationary, the initial memory can be computed with minimal error through the following integral:

$$\mathbf{y}(0) = \mathbf{E}(\boldsymbol{a}_b^\star) \int_{-\tau_{\min}}^{0} e^{\mathbf{\Lambda}\tau} \mathrm{d}\tau$$

where $\boldsymbol{a}_b^\star = \mathbf{V}_r^\mathsf{T} \mathbf{u}_b$ are the reduced coordinates of the base flow and $\tau_{\min}$ is the longest time horizon defined by the $\mathbf{\Lambda}$ matrix as in Equation (4.13). Finally, the parameters of the models are progressively optimised to reproduce the true transition trajectory as shown in Figure 4.13.

The CD-ROM model is integrated in time using an adaptive RK-45 scheme. Simulation results on the training trajectory are presented on Figure 4.14. It can be seen that the corrected model follows the training trajectory perfectly, triggering the transition at the right time, and correcting the oscillations of the original ROM during the stabilisation on the limit cycle. Moreover, the graph shows that the correction applied by the residual model is strong during the transition, where the original ROM struggles, and becomes minimal during the rest of the trajectory.

Finally, we present results of the performance of the model on the test trajectory. The first 8 seconds of DNS simulation are used to initialise the memory following Equation (4.14). Figure 4.15 presents the performance of each model on this trajectory. Because the starting point is not close to the base flow, the

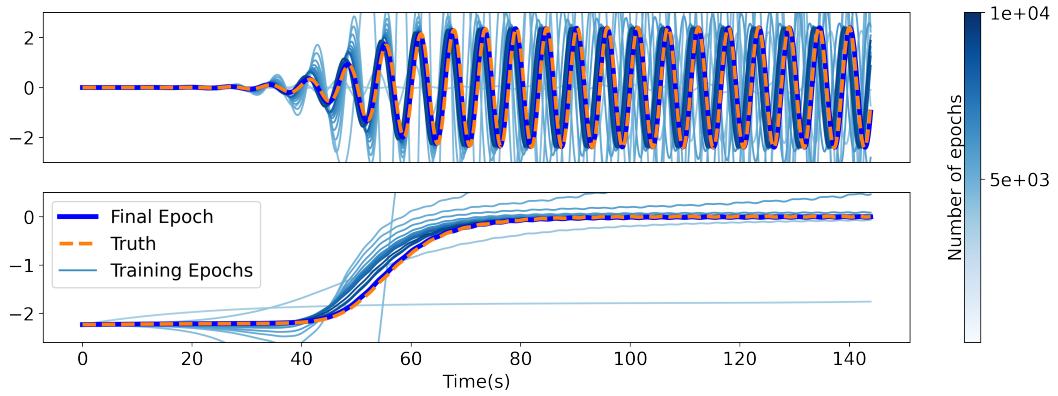


Figure 4.13: CD-ROM trajectory obtained at different levels of training. The first mode's amplitude is displayed in the top panel, while the third mode's amplitude is shown in the bottom panel.

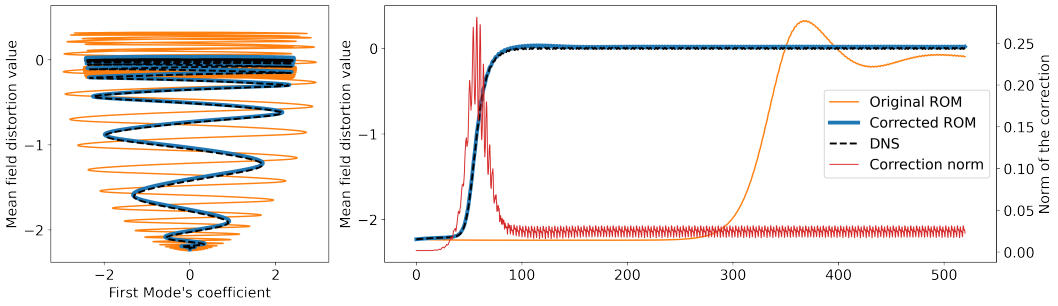


Figure 4.14: Results obtained with the corrected ROM for the simulation of the cylinder flow's transient dynamics. Left: phase space trajectory from base flow to limit cycle simulated with the finite element solver, the original Galerkin ROM and our corrected model. Right: time evolution of the shift mode's coefficient simulated with the same models, as well as the norm of the correction applied by our model.

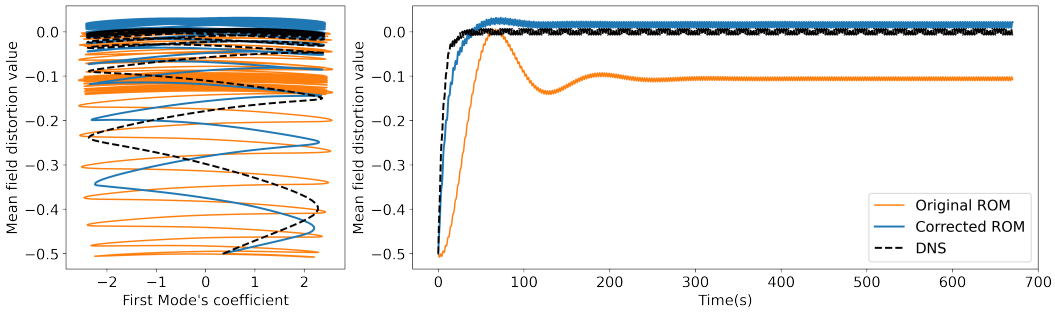


Figure 4.15: Simulation results obtained on a test trajectory. As in Figure 4.14, the first graph represents the phase space trajectory, while the time evolution of the shift mode's coefficient is presented on the second graph.

uncorrected ROM instantly exhibits transient dynamics, however, the growth rate of the transition is still inaccurate and the shift mode's trajectory presents the same non-physical oscillations around the limit cycle observed in the training trajectory. The corrected model does much better than the original ROM, simulating a more accurate transition, and stabilising on the limit cycle almost perfectly.

4.6.2 Fluidic pinball results

Correction Results

While the cylinder case discussed above offers a simple test bed for the presentation of the approach and its potential, it has already been shown that very parsimonious models could be used to model its dynamics [105], which suggests this configuration might not require a high dimensional neural network to learn a correction term. The case of the fluidic pinball is more challenging and can better underline the ability of our method to handle complex physics. As presented in the previous Section, the POD-Galerkin approach is not well suited to the reduction of the pinball case. The number of modes required to reconstruct the snapshot data with a satisfying accuracy is very large and using a small number of modes leads to an unstable model.

To apply the correction approach to the pinball case, a correction model is built. The encoder and residual models are multi layer perceptrons and the Sigmoid Linear Unit activation function is used as it leads to smoother integration. The weights of the residual model are initialised to be close to 0 so that the ROM is initially almost uncorrected. Diagonal entries of the memory matrix are initialised as a log-linear range of time horizons, ranging from 0.6 to 3.84 seconds. The training data consists of 1800 seconds (15000 snapshots) of DNS simulation in the chaotic regime. The leading two thirds of the simulated DNS trajectory are used for training while the remaining third is set aside for testing.

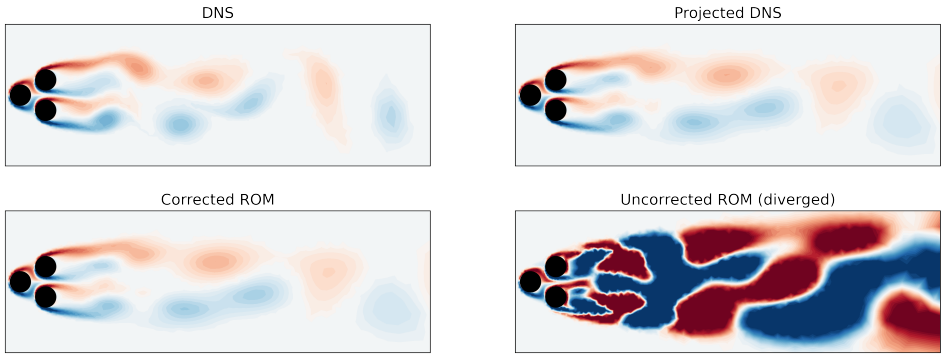


Figure 4.16: Vorticity fields obtained after 110 s of simulation in the chaotic regime.

As presented in the previous Section, the uncorrected 10-mode ROM is unstable and diverges after 70 s of simulation. As a result, trying to optimise the correction model for long trajectories directly would lead to a very unstable training process. To address this issue and stabilise the training, the corrected ROM is trained on sub-trajectories of only 10 seconds This length is chosen as it is short enough for the model to remain stable and long enough for the impact of the encoder on the memory to be accounted for. Once a good correction model has been trained for 10 seconds long trajectories, the length is progressively augmented to attain the target horizon of 120 s. Besides the stabilisation of the training, using sub-trajectories also allows for parallel training. Multiple sub-trajectories can be sampled from the snapshot data and integrated in parallel on a GPU to dramatically speed-up training, more information about the training strategy as well as the training parameters used is provided in 9.1.

The pinball correction model was trained to follow true trajectories up to 120 seconds. Results of the simulated flow fields are presented on Figure 4.16. It can be seen that the projection on the 10-dimensional POD basis effectively filters part of the spatial structures, leading to the divergence of the uncorrected reduced model. In contrast, the corrected ROM is able to reproduce the projected flow field accurately.

As with the cylinder case, the CD-ROM is simulated with an adaptive RK-45 scheme. Trajectory results simulated from a condition in the training basis are presented on Figure 4.17. The model starts quickly diverging from the true trajectory after the training horizon (120 s) as is expected from the chaotic nature of the problem. Deriving a model to perfectly follow the DNS trajectory would here make little modeling sense.

More interesting is the fact that, despite leaving the training trajectory, the corrected model does not become unstable, even when integrated for over 1000 seconds with an initial condition outside of the training data. This suggests that the dynamics correction learned by the model has some physical consistency,

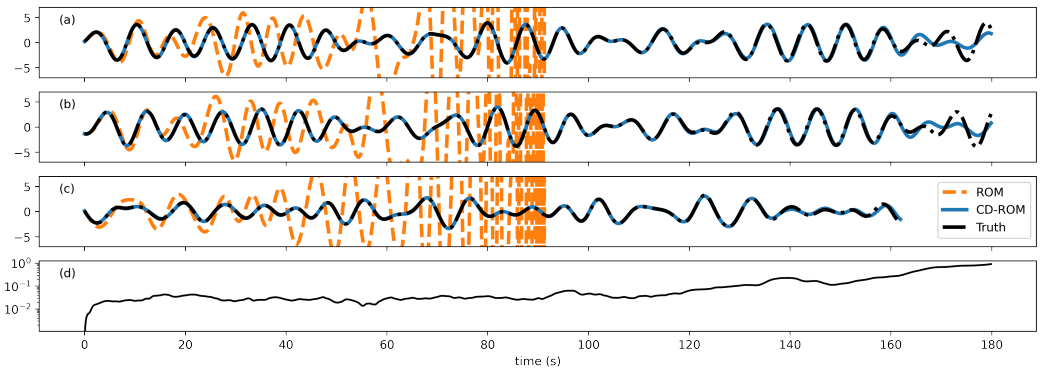


Figure 4.17: Results of the correction approach applied to the 10-mode ROM of the fluidic pinball. The initial condition is a random point selected in the training trajectory. Plots (a),(b),(c) describe the coefficients of the first three POD modes simulated with different models. Plot (d) presents the relative Euclidean distance between the true data and the trajectory simulated with the CD-ROM.

dissipating the necessary energy which would otherwise have made the simulation diverge in the uncorrected case.

The intuition that the model was able to learn a Physics-compatible correction is confirmed when looking at the statistics of the attractor spanned by the CD-ROM's trajectory. Using the `nolitsa` library [107], the correlation dimension ([13]) was estimated, as well as the maximum Lyapunov exponent ([14]) of the corrected and true trajectories. The results are shown in Figure 4.18 where the model is seen to reproduce well the characteristics of the true attractor. One can also look at the probability distributions of the mode's amplitudes, presented on Figure 4.19. Once again, the simulated trajectory reproduces the results of the true simulation.

Note that long trajectories (several hundreds of seconds) were simulated with the corrected model to obtain these statistics. Due to the chaotic nature of the problem and the length of the integration period, the model has visited parts of the attractor different from those seen in the training trajectory. Despite this, the CD-ROM remains stable and describes an attractor with statistics that are very similar to those of the true attractor, further supporting the approach for the reduced modeling of complex dynamics.

4.6.3 Parametric KS equation results

In this Paragraph, we present the results obtained by applying the CD-ROM architecture to a parametric case, the KS equation presented in Section 4.5.3. To do so, the reduced Galerkin model of equation 4.32 is augmented with the proposed CD-ROM architecture (equation 4.8). To account for the parametric nature of the problem, the coefficient ν is passed to both the residual ($\mathcal{R}$) and

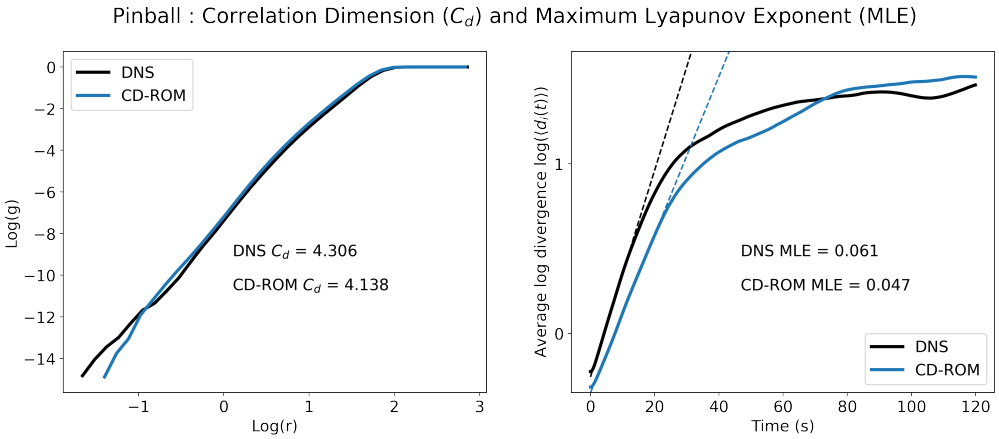


Figure 4.18: Statistics of the 10-dimensional attractors described by the projected DNS data (black) and the CD-ROM (blue). Left: Average number of points g in a sphere of radius r, the data follows the law $g = r^{C_d}$ where C_d corresponds to the correlation dimension of the attractor. Right: Time evolution of the average distance d between trajectories starting from arbitrarily close initial conditions on the attractor. The data follows the law $d = e^{tl_e}$ where l_e corresponds to the maximum lyapunov exponent of the system at hand.

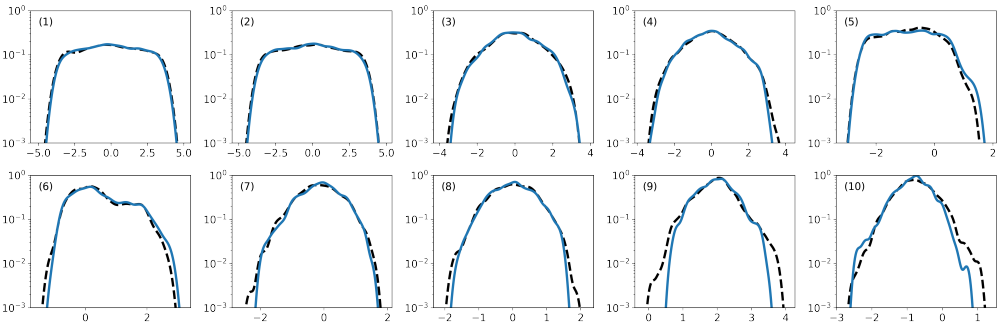


Figure 4.19: Estimated probability density functions for the coefficient of each mode. *Plain blue line*: Statistics of the trajectory simulated with the corrected model; *dotted black line*: Statistics of the projected DNS data. Labels refer to the mode index.

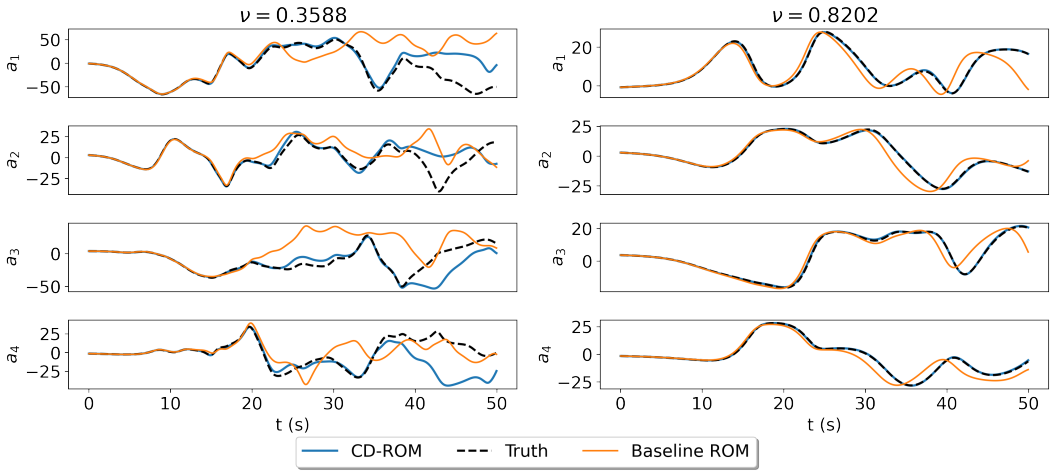


Figure 4.20: Coefficients of the first 4 POD modes simulated with the CD-ROM as well as the uncorrected Galerkin ROM under different parameter values in the test dataset. *Dashed line*: Projected DNS data, *full orange line*: Simulation of the Galerkin ROM, *full blue line*: Simulation of the CD-ROM.

encoder ($\mathbf{E}$) models, yielding the following CD-ROM system:

$$\begin{array}{rcccc} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a} & = & -\frac{1}{2}\boldsymbol{a}^{\mathsf{T}}\tilde{\mathcal{Q}}\boldsymbol{a} - \tilde{\mathcal{L}}\boldsymbol{a} - \nu\tilde{\mathcal{L}}^{2}\boldsymbol{a} & + & \mathcal{R}(\mathbf{y},\nu;\boldsymbol{\theta}_{\mathcal{R}}), \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y} & = & \mathbf{E}(\boldsymbol{a},\nu;\boldsymbol{\theta}_{\mathbf{E}}) & - & \boldsymbol{\Lambda}\,\mathbf{y}. \end{array} \tag{4.36}$$

The residual and encoder models are both expressed as multi layer perceptrons, using the SiLU activation function. The weights of both neural networks, as well as the memory matrix $\boldsymbol{\Lambda}$, are optimised using the Adam optimizer. As in the previous fluidic pinball case, we start by optimising the model on small sub-trajectories, then gradually increase the length of the sub-trajectories as the model reaches the desired accuracy. The model is trained on the data generated to computed the POD modes and assemble the Galerkin ROM (see Section 4.5.3). This training data corresponds to simulations carried out under 25 different parameter values in the range [0.3,1.5]. After training, the model is tested on 62 new simulations carried out under different parameter values selected randomly following a log-uniform distribution, as described in Section 4.5.3.

Contrary to the two previous flow cases, it is more efficient to simulate the KS equations using the semi implicit time-stepping scheme of Kar [39], thus we use this scheme to integrate the CD-ROM model in time. Figure 4.20 presents the results obtained by simulating the CD-ROM using ν values not included in the training data. On the two cases presented, the CD-ROM performs better than the Baseline Galerkin ROM. It can be seen that the model diverges from the true trajectory data earlier in the case where the value of ν is lower. This is expected as we showed in Section 4.5.3 that lower ν values lead to more complex dynamics.

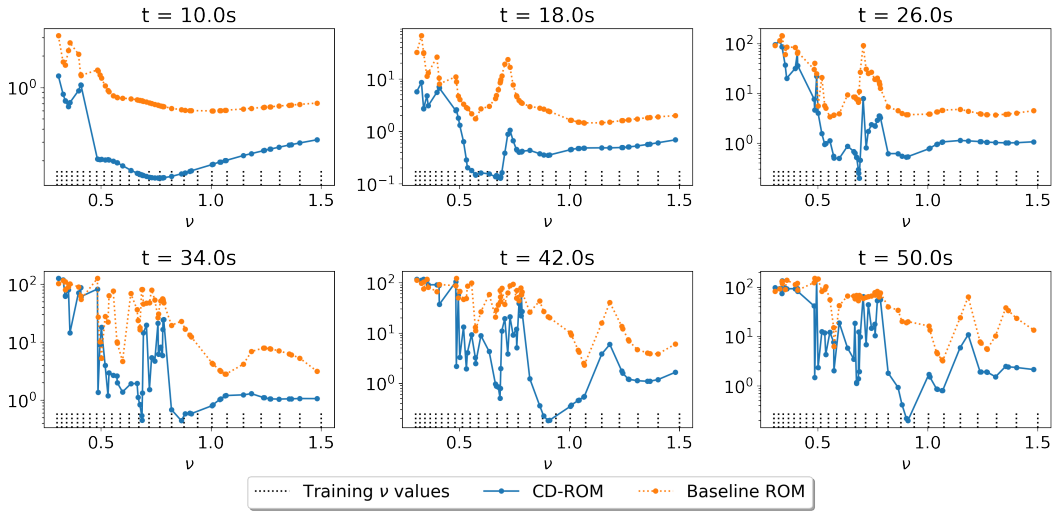


Figure 4.21: Error metric (equation 4.37) computed for each test parameter value.

To assess the performance of the CD-ROM over the whole test set, the Euclidean distance between the simulated reduced coordinates vector $\boldsymbol{a}$ and the projected DNS data $\boldsymbol{a}^\star$ is computed over time, and for every test parameter values:

$$d(t, \nu) = \|\boldsymbol{a}(t, \nu) - \boldsymbol{a}^\star(t, \nu)\|_2. \tag{4.37}$$

Figure 4.21 presents the values of the error metric (equation 4.37) for every test parameter value at select time steps. The figure shows that the CD-ROM is able to remain significantly closer to the true trajectory than the baseline Galerkin ROM for almost 20 seconds. The model then behaves differently depending on the parameter value. Cases in the $\nu \in [0.3, 0.5]$ range presenting the more chaotic dynamics quickly diverge from the true trajectory, while the CD-ROM is able to beat the baseline on the rest of the test cases for up to 50 seconds.

These results demonstrate the ability of the CD-ROM architecture to improve the Galerkin model order reduction approach in a parametric setting. The trained CD-ROM model is able to reproduce the dynamics of the full order system better than its uncorrected counterpart, even when using parameter values different from the training conditions.

4.7 Additional Results

4.7.1 Number of modes for the fluidic pinball reduction

Before concluding this Chapter, we discuss below some additional studies that were conducted on the case of the fluidic pinball. A first study considers the evolution of the uncorrected model error depending on the number of modes, which was carried out to underline the challenge of using only the ten leading modes in the experiments presented in the previous Section. A discussion on the computational costs of the method is then provided, to underline the parameters driving these costs as well as the tradeoff between the dimension of the ROM and the computational cost of the correction model.

Uncorrected models comparison

In Section 4.5, we present a 10-mode reduced order model of the fluidic pinball problem. The number of modes is chosen somewhat arbitrarily to challenge the method. The more modes are used to model the flow, the better the model will be at reproducing the true dynamics, reducing the complexity of the required residual term. With this study, we provide additional insights into the impact of the number of modes on the reduction problem to clarify the choice of using 10 modes to model the pinball flow.

Figure 4.22 presents the performance of different reduced models of the pinball flow. It is clearly seen that increasing the number of modes is beneficial for the performance of the reduced models. The magnitude of the closure term, as well as the speed at which the reduced model diverges from the true trajectory, are reduced when the number of modes increases. This Figure also shows that a higher number of modes leads to more stable reduced models. However, further experiments showed that even well resolved models such as the one using 173 modes were not stable and would diverge in certain conditions.

Training convergence

To add to the argument, a comparison of the training convergence between two models using different number of modes is discussed. Two models are built, using respectively 10 and 55 POD modes, and the correction architecture is trained using the same parameters, presented in Table 4.2.

Memory Dimension	$5 \times$ POD dimension
Residual lr	10^{-3}
Encoder lr	10^{-3}
Λ lr	$2\,10^{-4}$
Optimiser	AdamW

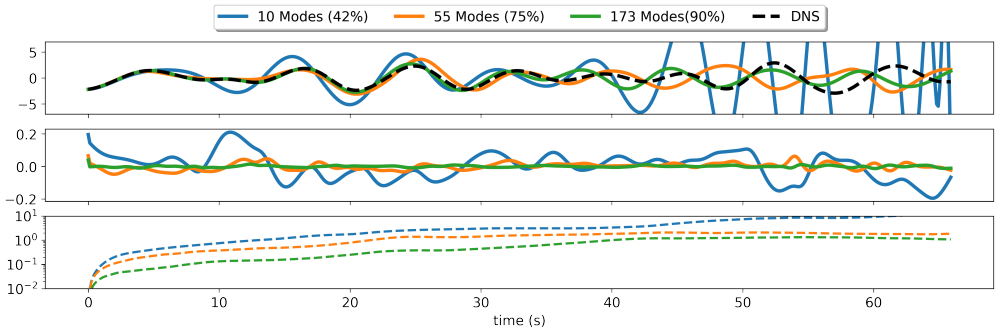


Figure 4.22: Comparison of the performance of POD-Galerkin models at different degrees of reduction. *Top*: Simulated value for the amplitude of the first POD mode; *center*: Value of the closure term for the first POD mode computed on the true trajectory; *bottom*: Relative distance between simulated and true trajectories.

Table 4.2: Parameters, e.g. learning rates (lr), used for the training of the compared CD-ROM architectures.

In both cases, the same loss is used, combining the optimisation of the distance between simulated and true trajectory with the residual regularization discussed in Section 4.3.4:

$$J = \frac{1}{r}\left(\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{a}(i\Delta_t) - \boldsymbol{a}^\star(i\Delta_t)\|_2^2 + c\frac{1}{n_t}\sum_{i=1}^{n_t}\|\mathcal{R}(\mathbf{y}(i\Delta_t)) - \mathcal{R}^\star(i\Delta_t)\|_2^2\right). \quad (4.38)$$

where $c = 0.1$ is a constant weighting the importance of the stochastic residual regularization term *w.r.t.* the trajectory loss. Notice that, to ease the comparison between the two models, the loss is scaled by the number of modes used in the ROM. The two models are trained in the same fashion, sub-trajectories of a hundred time steps are sampled in the training base and the loss (4.38) is optimised until a chosen threshold (5×10^{-4}) is reached, at which point the length of the sub-trajectories is increased by fifty time steps. This process is repeated until the model is able to reproduce sub-trajectories of a thousand time steps.

Figure 4.23 presents the evolution of the loss for the two models, as expected, the 55-mode model is quicker to train as it first reaches the threshold in 1400 epochs, while it takes the 10-mode model more than twice the number of epochs to achieve the same performance. Similarly, the 55-mode model achieves the required precision on trajectories of a 1000 time steps in only 4700 epochs, which is again more than twice as fast as the 10-mode model.

These results confirm the interest of only using the first 10 modes to challenge our correction approach. The relatively high magnitude of the residual term to be learned, the instability embedded in the model and the low degree

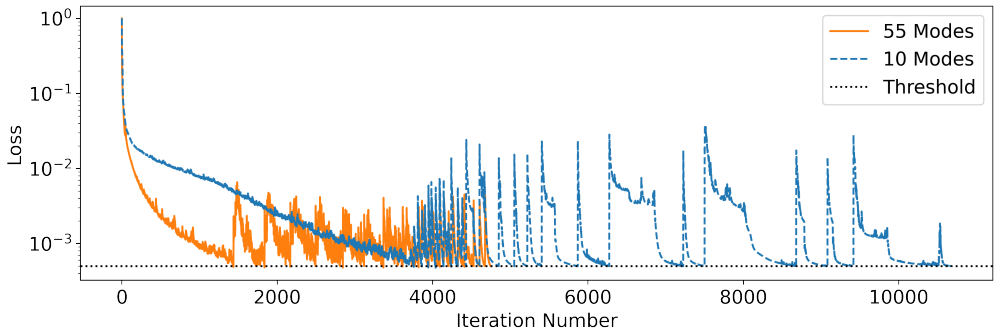


Figure 4.23: Loss evolution of two CD-ROM architectures using a different number of modes.

of resolution of the problem (only 42% of the snapshot information) all constitute significant complexities which could arise in real world applications. While we showed that using a higher number of modes would simplify the modeling problems, this constraining choice helps demonstrate the applicability of the CD-ROM method to challenging modeling problems.

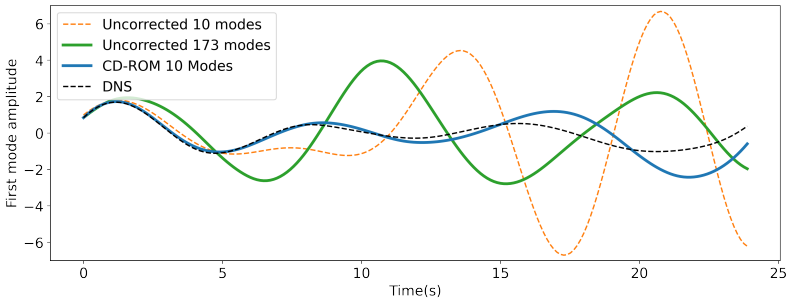
4.7.2 Computational cost

While the computational cost of the overall CD-ROM approach and the way it compares to full order methods will strongly vary with the nature of the problem it is applied to, the simulation software used as well as the available hardware, we provide some elements of comparison with the POD Galerkin method in this Section. Once again focusing on the case of the fluidic pinball (see Section 4.5.2), we distinguish several components of the computational costs entailed by the CD-ROM method:

ROM assembly

Because most POD Galerkin models are often restricted to a very low number of modes, the cost of assembling the reduced model can often be overlooked. However, some problems might require a high number of POD modes to achieve a satisfactory resolution. For example, the fluidic pinball case requires up to a thousand modes to capture 99% of the snapshot information, which directly impacts the cost of assembling the reduced operators $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{Q}}$ in equation 4.1. Specifically, the reduced advective operator $\tilde{\mathcal{Q}}$ requires the computation of $O(n^3)$ inner products, n being the number of selected POD modes.

This leads to exploding ROM assembly costs as the number of POD modes grows higher. While the assembly remains a one-time, parallelisable operation, we observed that assembling a 250 modes ROM on a 50 cpu machine took more than a full day of computation. This underlines the interest of representing



Integration times	
Uncorrected 10 modes	30 ms
Uncorrected 173 modes	550 ms
CD-ROM 10 modes	330 ms

Figure 4.24: Performance and computational cost of simulating different models starting from an initial condition outside the CD-ROM training basis.

the dynamics on a low dimensional basis of modes, as assembling a thousand mode ROM would become prohibitively expensive.

ROM Simulation

Figure 4.24 presents the comparative simulation costs and performance of different reduced models on the fluidic pinball case. The figure shows that, although the CD-ROM does diverge from the true trajectory after some time, it performs better than its uncorrected counterparts. Specifically, the uncorrected 173 modes reduced model which captures more than 90% of the snapshot information, diverges earlier than the CD-ROM, while being more expensive to simulate.

It can also be seen that the CD-ROM is significantly more expensive to simulate than the simple Galerkin model. This can be explained by the cost of evaluating the neural networks embedded in the CD-ROM architecture. Indeed, neural networks require the evaluation of matrix vector products of relatively high dimension. In the pinball case, we use two hidden layers of 250 neurons for the correction model, which explains the computational cost increase. However, it is interesting to note that the cost of evaluating a multi layer perceptron scales quadratically with its width (number of neurons per layer), while the cost of evaluating the advection term in the galerkin ROM scales cubically with the number of modes. This explains the fact that the 173 modes Galerkin ROM is more expensive to simulate than the 10 dimensional CD-ROM, while having a lower accuracy. Similar to the previous Paragraph, this shows the interest of correcting a low dimensional model, rather than simply increasing the number of modes.

Notes on Full Order Models

Providing a detailed comparison with full order methods is outside of the scope of this study, as the computational cost of a full order model will depend on a large number of choices, ranging from simulation software implementation to numerical integration choices. We can however state that in fluid mechanics examples, the full-order models were extremely expensive to simulate when compared to the reduced models studied in this Chapter. For example, generating the snapshot and test data for the fluidic pinball case took more than a day on a 50 cores machine. By comparison, the simulation of the trained CD-ROM for the same duration is of the order of the second on a normal computer.

4.8 Conclusion

This Chapter introduces an augmented reduced order modeling strategy based on the *hybridization* of the classical Galerkin projection method and simple neural networks. By studying the limitations of the Galerkin projection, we establish links with the theory of partially observed systems, which leads us to use past observables of the studied system as a critical ingredient for the correction of Galerkin models. Building on this result, the CD-ROM architecture is proposed to extract and exploit useful information from the system trajectory, by embedding the model with a delay differential equation structure. Moreover, the training strategy based on adjoint optimization ensures *a-posteriori* performance of the model on the training trajectory.

The CD-ROM approach was demonstrated on two simple CFD test cases, namely, the flow over a cylinder and the fluidic pinball. Numerical experiments have shown that the corrected models were able to capture the true dynamics with a high degree of accuracy, reproducing the true transition in the case of the cylinder flow, and following the training trajectory for multiple Lyapunov times in the fluidic pinball case. Moreover, these experiments outlined the reliability of the corrected model as it performed better than its uncorrected counterpart even outside the training conditions. The results obtained with the fluidic pinball are particularly encouraging. We showed that the correction model was able to stabilize the original Galerkin model in a consistent physical manner, as the attractor simulated with the CD-ROM approach presents statistics similar to the original attractor.

The ability of the proposed approach to extend to parametric problems was also demonstrated. The CD-ROM method was applied to the case of the Kuramoto-Sivashinsky equation with varying viscosity. After training the model on a small number of parameter values in a selected range, we showed that the CD-ROM approach improved the performance of the baseline Galerkin ROM over the whole parameter range, even when simulating using parameter values

outside of the training data. This is of interest for many real-world situations, e.g. industrial applications where a low-cost parametric model is a key enabler.

With this, we conclude this Chapter on the CD-ROM method. We once again emphasize the *hybrid* nature of the model, combining first principles and neural modeling. We show in the following Chapter that the degree of *hybridization* of the model can be adjusted to allow for the reduction of challenging non-linear dynamics. We also show that the time-continuous and interpretable memory formulation of the CD-ROM learns to select frequencies that are consistent with the system of interest.

CHAPTER 5

ADAPTIVE HYBRIDIZATION AND MEMORY INTERPRETABILITY

Contents

5.1 Introduction 93

5.2 Application to Calendering and adaptive hybridization 94

5.2.1 Problem introduction 95

5.2.2 Proposed modeling approach 96

5.2.3 Results 98

5.2.4 Study Conclusion 100

5.3 Memory interpretability and Time horizons . . . 102

5.3.1 Introduction 102

5.3.2 Memory *Horizons* 102

5.3.3 Experiment setup 103

5.3.4 Modeling 104

5.3.5 Results 104

5.4 Conclusion 106

5.1 Introduction

The previous Chapter introduced the CD-ROM method for the augmentation of POD-Galerkin reduced-order models. The method was shown to be able to

correct imperfect reduced order models resulting from the reduction of a system's governing equation on a low dimensional linear basis. In this Section, we discuss additional studies carried out with the CD-ROM method.

First, we introduce results that were presented in Menier et al. [176]. This work was carried out in collaboration with *Michelin R&D* in the context of the HSA project at IRT SystemX. It focuses on the reduced-order modeling of an industrial process with the CD-ROM method, the calendering of tire rubber and illustrates the adaptability of the approach to previously irreducible equations.

A second study is then presented on the time horizons learned by the CD-ROM architecture. By analyzing the memory dynamics of the CD-ROM, we show that the model is able to choose and extract the dominating frequencies from system data.

5.2 Application to Calendering and adaptive hybridization

In this Section, we apply the CD-ROM approach to an industrial modeling problem. We show that the approach provides an adjustable degree of intrusivity as it can be used to learn both a closure model as well as specific terms in partial differential equations. Thus, we can consider this application of the CD-ROM to be a relatively more data-driven model than the original approach proposed in the previous Chapter. Similar to the native CD-ROM approach presented in the previous Chapter, we place the method proposed in this Section on the *hybridisation spectrum*,displayed on figure 5.1.

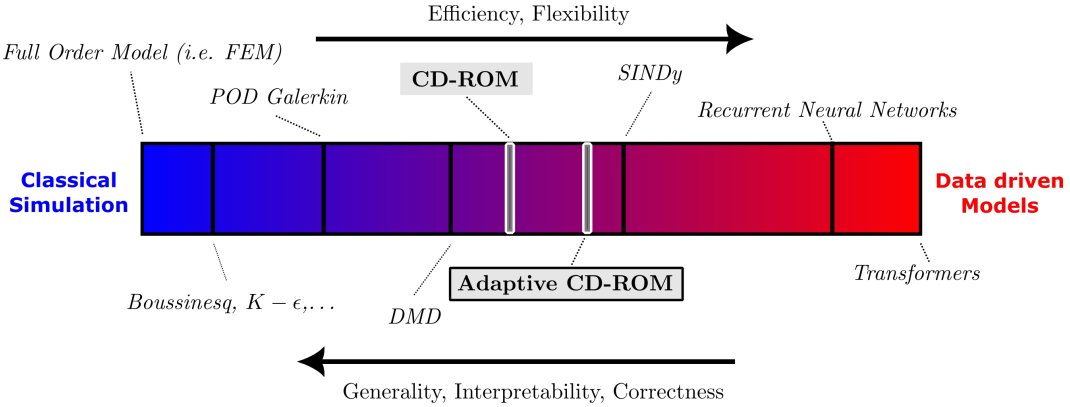


Figure 5.1: The adaptive CD-ROM approach on the hybridization spectrum. *NB: This figure is provided for illustrative purposes and helps frame our proposals within the broader context of the thesis. The placement of each method on the spectrum is open to discussion, and small variations could lead to displacing a given approach closer to one end or the other.*

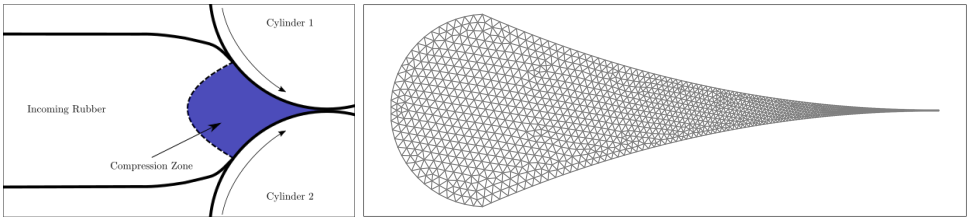


Figure 5.2: *Left*: Schematic of the calendering process. *Right*: Simulation mesh used in the finite element solver.

By augmenting an incomplete reduced order model of the system of interest with the CD-ROM architecture, we show that an efficient and flexible model is obtained, able to accurately simulate the problem in conditions unseen during training and linearly reduce previously *irreducible* dynamics.

5.2.1 Problem introduction

Industrial problem: Rubber calendering process

This study focuses on the modeling of the rubber calendering process. Calendering is a manufacturing process that consists in passing a rubber sheet between two rollers to obtain the desired thickness and mechanical properties (see Figure 5.2). Because of the compression between the rotating cylinders, the rubber can heat up and deteriorate. To address this issue, one needs to estimate the heat generation inside the material under different cylinder rotation speeds in order to determine acceptable process conditions. The issue is that simulating the problem in a classical finite elements solver can take too much time, limiting the applicability of full order simulation approaches to the control of the process. We hence propose to use model order reduction to lower the cost of simulating the problem.

Governing equations

The dynamics of this problem are governed by the following system of partial differential equations:

$$0 = -\nabla(2\eta(\mathbf{u}, T)\gamma(\mathbf{u})) - p, \tag{5.1}$$

$$\frac{\partial T}{\partial t} = \frac{\lambda}{\rho C_p}\nabla^2 T - \mathbf{u}\cdot\nabla T + \frac{\eta(\mathbf{u}, T)\gamma(\mathbf{u})^2}{\rho C_p}. \tag{5.2}$$

where $T(x,t)$ represents the value of the temperature of the rubber, $\mathbf{u}(x,t)$ is the velocity of the rubber, p the pressure in the system, η the dynamic viscosity of the material and γ the deformation rate. Note that although the velocity of

the fluid is critical in the modeling of the system as it captures the effects of the rotating cylinders on the system, it is governed by a steady-state equation. This is because in this case, the velocity field of the rubber reacts rapidly and is considered to transition instantly to its steady state under the given cylinder speed and temperature conditions. The second equation then represents the dynamics of the temperature field as it reacts to the velocity field imposed by the cylinders.

Building a reduced-order model of these equations using the POD-Galerkin method is not straightforward. First, the steady-state problem in Eq.(5.1) is unsuitable for the efficient simulation of this system, as it has to be solved at every time step of the integration. This requires additional considerations on the choice of solver, and can lead to expensive and unstable integration.

Second, Eq.(5.2) can not be efficiently reduced through linear projection. Because of the linear nature of the Laplacian and gradient operators, the first two terms of the dynamics can directly be projected on a linear low dimensional basis of spatial modes, reducing their computation to simple tensorial operations as shown in Section 3.3. The last term, however, is a source term accounting for the heat generated by the deformation of the rubber. This phenomenon is strongly nonlinear and cannot be reduced linearly. Computing its reduced form would require back-and-forth exchanges between the full-order solver and the reduced model, which would directly impact the computational performances of the ROM. To avoid these costly steps, we extend the CD-ROM approach to model the last term of Eq. (5.2) in addition to the required correction term.

5.2.2 Proposed modeling approach

Data Generation

To construct the ROM, we first assemble a collection of solutions of the system at different time steps and under different cylinder rotational speeds ($\mathcal{S}(t)$) by simulating Eq. (5.2) with the finite element solver MEF++[1] [144]. To generate the data, cylinder speed trajectories are sampled from the following distribution:

$$\mathcal{S}(t) = c_0 + \sum_{i=4}^{7} c_i \sin\left(\frac{2\pi t}{2^i}\right), \qquad c_0 \sim \mathcal{N}(1, 0.25), \quad c_i \sim \mathcal{N}(0, 0.25). \tag{5.3}$$

This distribution yields cylinder speed trajectories in a range representative of the operating conditions of the calendering process. 20 trajectories of $\mathcal{S}(t)$ are sampled from this distribution, under which the system is simulated to generate solution snapshots.

[1] MEF++ — Wikipédia, http://fr.wikipedia.org/w/index.php?title=MEF%2B%2B&oldid=192108614

ROM

By computing the Proper Orthogonal Decomposition of the obtained snapshots as in Section 2.2.2, we extract a reduced number of principal modes optimally approximating the data. Following this strategy, we compute two orthonormal bases of modes, $V_T \in \mathbb{R}^{n_c \times n_T}$ for the temperature and $V_{\mathbf{u}} \in \mathbb{R}^{2n_c \times n_{\mathbf{u}}}$ for the velocity. Here n_c denotes the number of grid cells in the mesh, n_T the number of selected temperatures modes, and $n_{\mathbf{u}}$ the number of selected velocity modes, so that each column of the matrices V_T and $V_{\mathbf{u}}$ represents a complete field. After computing these two bases, one can approximate the solution as linear combinations of the principal components: $\widetilde{T} = V_T \boldsymbol{a}_T$ and $\widetilde{\mathbf{u}} = V_{\mathbf{u}} \boldsymbol{a}_{\mathbf{u}}$. With this formulation, solving the problem reduces to computing the low dimensional vectors of POD coordinates $\boldsymbol{a}_T \in \mathbb{R}^{n_T}$ and $\boldsymbol{a}_{\mathbf{u}} \in \mathbb{R}^{n_{\mathbf{u}}}$.

Observing that the critical quantity to be modeled is the temperature field, and that the computation of the velocity field implies the resolution of a non-linear system of equations (Eq.(5.1)), we propose to model the reduced velocity coordinates as a function of the cylinder speed and temperature:

$$\widetilde{\boldsymbol{u}} \approx \widetilde{\boldsymbol{u}}(\mathcal{S}, \boldsymbol{a}_T). \tag{5.4}$$

The above equation is only an approximation as modeling the solution of the Stokes problem in Eq.(5.1) exactly might not be feasible. However, we note that any approximation error introduced by this modeling choice can be accounted for by the CD-ROM closure model. Thus, the problem is reduced to the modeling of the dynamics of the reduced temperature coordinates $\boldsymbol{a}_T$.

Following the POD-Galerkin method introduced in Section 3.3, the reduced forms of the temperature and velocity fields are injected in the temperature dynamics (Eq. (5.2)), which are then projected on the temperature POD basis V_T, yielding a system of n_T ordinary differential equations:

$$\begin{aligned} \frac{d\boldsymbol{a}_T}{dt} = & \underbrace{\frac{\lambda}{\rho C_p} V_T^{\mathsf{T}} \nabla^2 V_T \boldsymbol{a}_T - \boldsymbol{a}_{\mathbf{u}} V_T^{\mathsf{T}} (V_{\mathbf{u}} \cdot \nabla V_T) \boldsymbol{a}_T}_{\mathbf{r}(\boldsymbol{a}_T, \mathcal{S})} \\ & + \underbrace{V_T^{\mathsf{T}} \frac{\eta(V_{\mathbf{u}} \boldsymbol{a}_{\mathbf{u}}, V_T \boldsymbol{a}_T) \gamma(V_{\mathbf{u}} \boldsymbol{a}_{\mathbf{u}})^2}{\rho C_p} + \mathcal{R}}_{\mathbf{i}(\boldsymbol{a}_T, \mathcal{S})} \end{aligned} \tag{5.5}$$

Where $\mathcal{R}$ is the residual introduced in Section 3.3 that results from the evaluation of the dynamics from the approximate reconstructed states $\widetilde{\boldsymbol{u}}$ and $\widetilde{T}$. The above equation can be separated into two parts: a *reducible* part $\mathbf{r}$ which easily expresses in terms of the reduced coordinates $\boldsymbol{a}_{\boldsymbol{u}}, \boldsymbol{a}_T$, and an *irreducible* part $\mathbf{i}$ which cannot be directly evaluated in the reduced space. This would normally be a major impediment to the use of the POD-Galerkin method to solve this

problem. However, using the CD-ROM approach, we can learn the effect of $\mathbf{i}$ on the reduced dynamics using a neural network. Allowing for the extension of reduced order modeling approaches to previously irreducible problems, while retaining as much as possible from the original dynamical equations:

$$\frac{d\boldsymbol{a}_T}{dt}(t) = \mathbf{r}(\boldsymbol{a}_T, \boldsymbol{a}_\mathbf{u}, \mathcal{S}) + \mathcal{NN}(\boldsymbol{a}_T, \boldsymbol{a}_\mathbf{u}, \mathcal{S}, y) \tag{5.6}$$

$$y(t) = \int_{-\infty}^{t} e^{(s-t)\Lambda} x(s) \mathrm{d}s, \qquad x(t) = [\boldsymbol{a}_T(t), \mathcal{S}(t)] \tag{5.7}$$

where Λ is a positive diagonal matrix corresponding to the time horizon matrix defined in the previous Chapter (see Section 4.3.2) and $y(t)$ is the memory of the model, specifically designed to be continuously integrable in parallel with the reduced dynamics as a simple linear system. In the previous Chapter, the critical role of the memory term $y(t)$ in retrieving information necessary for the correction of reduced order models was underlined. In the following Section, we show that this same idea can be used to model both the residual $\mathcal{R}$ and the irreducible terms in the original equations.

5.2.3 Results

POD Bases

POD bases are computed for both the velocity and temperature fields. Figures 5.3 and 5.4 respectively present the results of the POD method on the temperature and velocity fields.

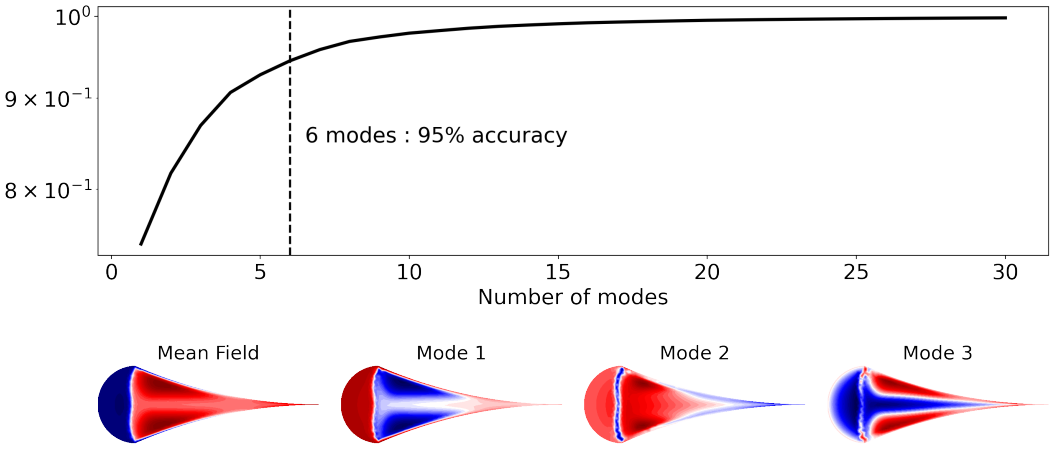


Figure 5.3: Temperature field POD. *Top:* Relative training data reconstruction error depending on the number of modes. *Bottom:* Visualization of the leading POD modes.

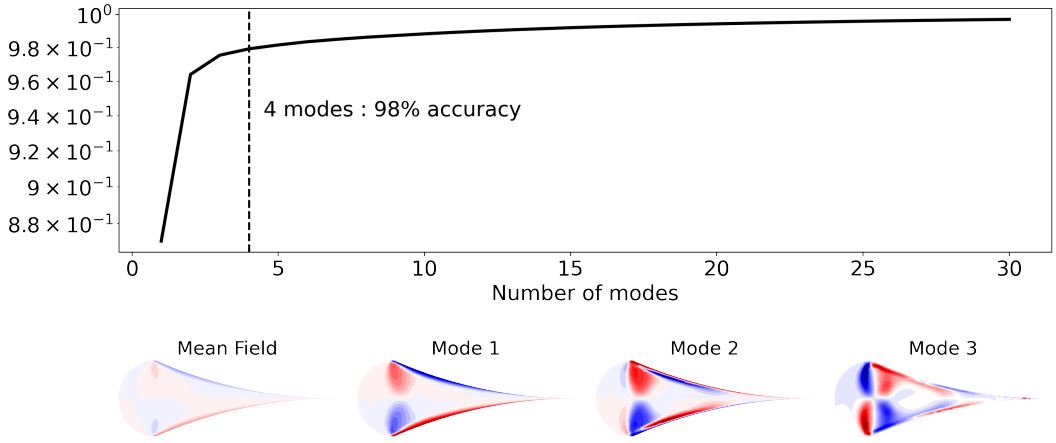


Figure 5.4: Velocity field POD. *Top:* Relative training data reconstruction error depending on the number of modes. *Bottom:* Visualization of the vertical component of the leading POD modes.

The results display clearly that both the temperature and velocity fields can be reduced on very low dimensional linear bases. We see that selecting the leading $n_T = 6$ POD modes captures more than 95% of the variance in the problem. Similarly, the leading $n_{\boldsymbol{u}} = 4$ velocity modes are enough to capture more than 98% of the variance in the velocity snapshots.

Velocity modeling

As mentioned in the previous Section, the equations governing the velocity field are not solved in the reduced model to avoid costly non-linear system resolutions. Instead, a simple machine learning model is chosen to learn the relation between the reduced velocity coordinates $\boldsymbol{a_u}$, the cylinder velocity $\mathcal{S}$ and temperature coordinates $\boldsymbol{a}_T$:

$$\hat{\boldsymbol{a}}_{\boldsymbol{u}} = f(\boldsymbol{a}_T, \mathcal{S}). \tag{5.8}$$

In this study, we defined f as a simple ridge regression model in order to reduce as much as possible the variance in the reduced part $\mathbf{r}$ of the model. Indeed, this reduced part constitutes the basis upon which the CD-ROM model is built, meaning that the inaccuracies introduced by this simple ridge regression model can be accounted for by the non-linear closure, $\mathbf{i} = \mathcal{NN}(\boldsymbol{a}_T, \hat{\boldsymbol{a}}_{\boldsymbol{u}}, \mathcal{S}, y)$.

Test performance

The model is trained on 80% of the trajectories in the dataset using the NeuralODE approach in combination with the Adaptive Checkpointing Adjoint method

[150]. The objective $\mathcal{L}$ is defined as the mean squared Euclidean distance between the simulated reduced coordinates and their true value $\boldsymbol{a}_T^\star$:

$$\mathcal{L} = \frac{1}{n_t + 1} \sum_{i=0}^{n_t} \|\boldsymbol{a}_T^\star(t_i) - \boldsymbol{a}_T(t_i)\|_2^2 \tag{5.9}$$

The remaining 20% of the dataset is used for testing. Figure 5.5 presents the performance of the model and its uncorrected counterpart on a test trajectory. We also compute the RMSE normalized by the standard deviation of the data (*NRMSE*) to provide a quantitative indication of the performance of the model on the test trajectories. Obtained results show that the CD-ROM trajectory fits the true trajectory almost perfectly, compared with the incomplete ROM (Figure 5.5). The final NRMSE computed over the complete test set is of 2.5% . Moreover, the simulation of the corrected reduced model is much more computationally efficient than finite element solvers as the parallel simulation of 128 trajectories only takes a few seconds on a RTX 2080 GPU, while the simulation of a single trajectory in our finite elements solver took about 5 minutes. Note that similarly to the values presented in Section 4.7, these simulation times only provide a rough estimation of the performance gap, as they heavily depend on the hardware, implementation and simulation parameters of both the ROM and the FE model.

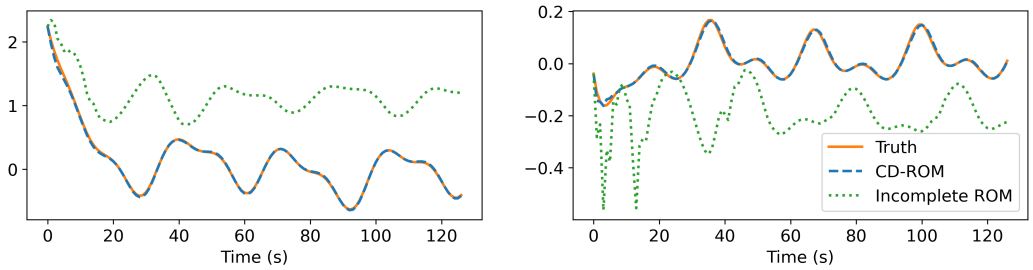


Figure 5.5: Performance of the corrected ROM (Eq. (5.6)) on a trajectory unseen during training. *Left*: Trajectory of the first mode a_1. *Right*: Trajectory of the second mode a_2. The incomplete ROM ($\mathbf{r}$ in Eq. (5.5)) is also shown for comparison.

5.2.4 Study Conclusion

This study presented an application of the CD-ROM method introduced in the previous Chapter to an industrial problem. We tried to underline the adaptability of the CD-ROM approach to ill-posed reduction problems such as those involving highly nonlinear terms, while retaining a high degree of interpretability compared to models unaware of the governing equations.

Moreover, we showed that the CD-ROM model can be used outside of its training conditions with a high degree of accuracy. In the future, the ability of

the model to generalize to new materials in addition to new conditions should be investigated. The proposed approach could then be used in model predictive control strategies to optimally tune the parameters of the manufacturing process.

5.3 Memory interpretability and Time horizons

5.3.1 Introduction

This Section presents a study of the ability of the CD-ROM to tune its memory mechanism to select and retain the useful frequencies in the history of a system. Indeed, the previous Chapter discussed the interpretability of the memory architecture of the CD-ROM, which is based on a simple low-pass filtering mechanism of the system trajectory.

In this short study, we show that the model is able to learn relevant memory *time horizons* to optimally retain the information necessary to the modeling of a system's dynamics. To do so, we discuss the notion of *time horizon* of the memory and then introduce a simple test case that allows for the study of the memory mechanism and its interpretability.

5.3.2 Memory *Horizons*

As presented in the previous Chapter, the memory of the CD-ROM, $\mathbf{y} \in \mathbb{R}^{d_y}$, evolves according to the following dynamics:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{E}(\mathbf{x};\theta) - \Lambda_\theta \mathbf{y}. \tag{5.10}$$

Where $\mathbf{x} \in \mathbb{R}^{d_x}$ is the state of the system, $\mathbf{E}(\cdot;\theta) : \mathbb{R}^{d_x} \mapsto \mathbb{R}^{d_y}$ is a trainable map that *lifts* the state to the memory space and $\Lambda_\theta \in \mathbb{R}_+^{d_y \times d_y}$ is a positive diagonal matrix. Because the matrix Λ is diagonal, the dynamics in memory space are uncoupled, thus, each dimension of the memory $\mathbf{y} = [y_1, y_2, \ldots, y_{d_y}]$ can be isolated and its value computed as follows:

$$y_i(t) = \int_{-\infty}^{t} e^{(s-t)\lambda_i} E_i(\mathbf{x}(s);\theta) ds. \tag{5.11}$$

Where λ_i is the diagonal entry of Λ corresponding to dimension i, and E_i the corresponding dimension of the output of $\mathbf{E}$. From the above equation, we define a *time horizon* τ_i for the information in the memory dimension y_i:

$$\tau_i = \frac{1}{\lambda_i}. \tag{5.12}$$

Thus τ_i corresponds to the length of time necessary for the importance of a given state $E_i(\mathbf{x}(\mathbf{s}))$ in memory to decrease by a factor of e^{-1}. Note that this quantity is also directly related to the cutoff frequency of the filter in Eq.(5.11). In the following Sections, we show that the CD-ROM approach systematically learns similar values for the time horizons τ_i starting from different initial conditions. We also show that these horizons are coherent with the system to be learned.

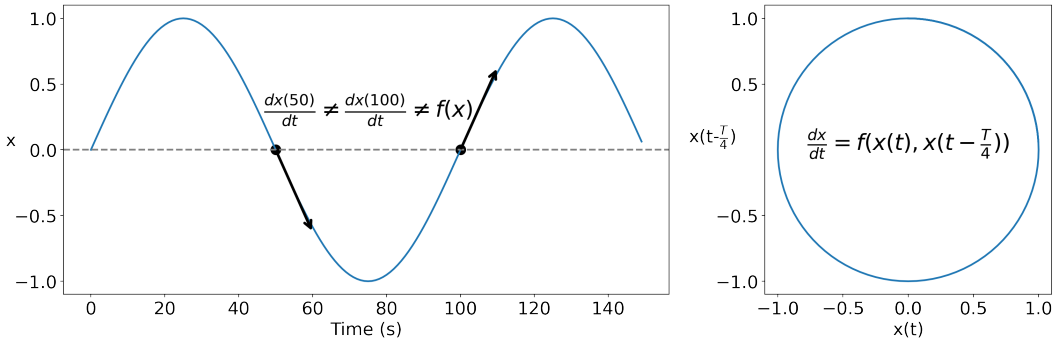


Figure 5.6: A single frequency sine wave results from a two-dimensional system, it can not be modeled through one-dimensional Markovian dynamics. However, a delayed coordinate can be used to model the dynamics as a non-markovian process.

5.3.3 Experiment setup

A simple test case

To analyze the way the CD-ROM's memory architecture selects time horizons, we chose the simplest non-markovian test case possible, a superposition of sine waves. Sine waves have the advantage of being driven by clearly identified frequencies, and despite their simplicity, they cannot be modeled as Markovian autonomous systems (see figure 5.6).

They can however be augmented with delayed coordinates to obtain a non-markovian dynamical model for the wave. We see in figure 5.6 that when the wave only has one frequency, taking a delay of $\frac{T}{4}$ with T the period of the wave yields a coordinate system in which the system is a simple limit cycle, and can thus be modeled directly as an autonomous system.

A similar idea is used in this study as we analyze how the CD-ROM model uses the memory dimensions, which play a similar role to delayed coordinates, to learn the non-markovian dynamics of a superposition of two waves. We simplify the memory architecture in Eq.(5.10) by defining the map $\mathbf{E}$ as a simple repetition of the state of the wave $x(t) \in \mathbb{R}$ so that:

$$\mathbf{E}(x) = [x \times d_y]. \tag{5.13}$$

This means that the dimensions of the memory $y(t)$ directly correspond to low-pass filtering of the trajectory $x(t)$. Finally, the trajectory to be learned is defined as the superposition of two waves of periods $T_1 = 50s$ and $T_2 = 168s$:

$$x(t) = sin(\frac{2\pi}{50}t) + \frac{1}{2}sin(\frac{2\pi}{168}t). \tag{5.14}$$

5.3.4 Modeling

The dynamical model is defined as follows:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \mathcal{NN}(x, \mathbf{y}; \theta) \tag{5.15}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \mathbf{E}(x) - \Lambda_\theta \mathbf{y}. \tag{5.16}$$

Where $\mathcal{NN}$ is a simple multi-layer perceptron mapping the delayed coordinates to the value of the dynamics, $\mathbf{E}$ and Λ are the operators introduced in equation 5.10, and the dimension of the memory is chosen to be $d_y = 3$, which is required as the wave is driven by two different frequencies.

The above model can be trained very easily to reproduce the trajectory of the sine wave in Eq.(5.14), following the training strategy introduced in Section 4.3.3.

5.3.5 Results

To gather information in memory, the model learns the diagonal entries λ_i of the memory matrix to select the relevant frequencies in the trajectory of the system. We confirmed this behavior by training the model starting from different initializations for the memory matrix. In each test, the entries of the memory matrix were initialized to be equal to the same value at the start of the training so that:

$$\frac{1}{\lambda_{1,\mathit{init}}} = \frac{1}{\lambda_{2,\mathit{init}}} = \frac{1}{\lambda_{3,\mathit{init}}} = \tau_0. \tag{5.17}$$

50 values for τ_0 are sampled logarithmically in the range $[20, 200]$. The results obtained after training the model with three of those values are presented on figure 5.7. We then present the distributions of the final values for the *time horizons* τ_i computed from the memory entries λ_i at the end of training with the aforementioned 50 different initial values τ_0 in figure 5.8.

Figure 5.8 clearly shows that the model converged to similar time horizons despite the strongly differing initializations. This result confirms the CD-ROM model's ability to adequately learn the matrix Λ entries irrespective of the initialization. It is also interesting to note that the time horizons τ_i are coherent with the system. The longest time horizon $\tau_3 \approx 163s$ is very close to the largest period in the data $T_2 = 168s$, similarly $\tau_2 \approx T_1 = 50s$, finally the shortest time horizon τ_1 is close to the optimal discrete delay value $\frac{T_1}{4}$.

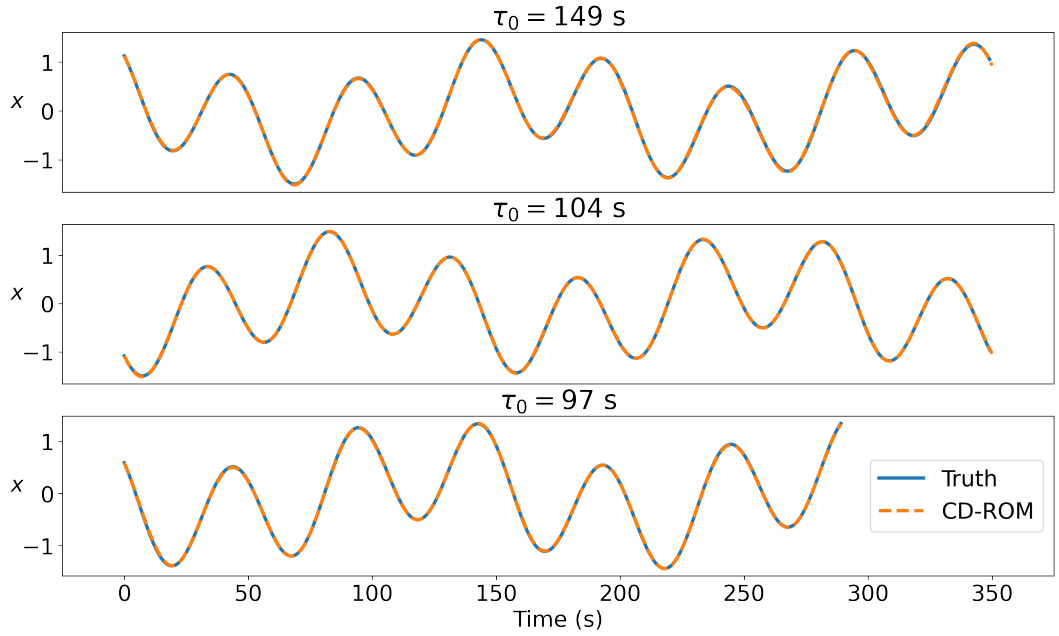


Figure 5.7: True and predicted values obtained after training the CD-ROM architecture in Eq.(5.15) with three different initial values τ_0 for the time horizons.

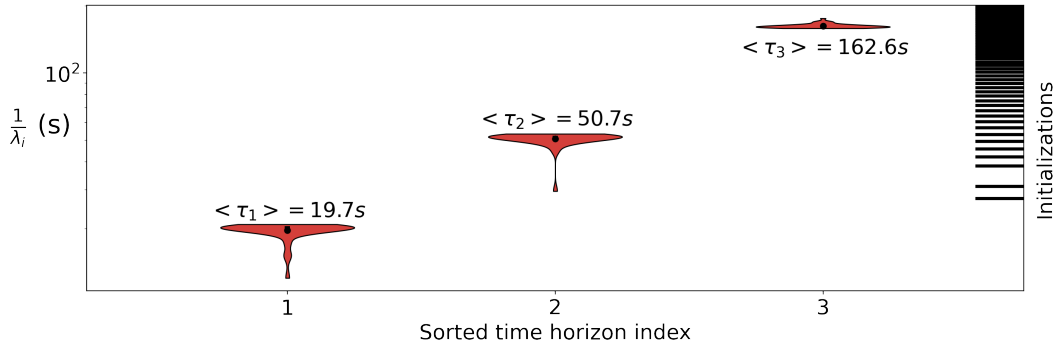


Figure 5.8: Distributions of the learned time horizons at the end of the training, starting from 50 constant initializations. Average values for the time horizons are denoted as $< \tau_i >$. Three cases did not lead to a converged model and are not presented here.

5.4 Conclusion

This Chapter introduced two additional studies carried out with the CD-ROM model. First, we presented an application of the CD-ROM method to an industrial problem. Showing that in addition to yielding satisfactory performance on the modeling of the calendering of tire rubber, the CD-ROM model could be extended to strongly nonlinear dynamics. Overcoming limitations of the POD Galerkin reduction method while retaining the part of the governing equations that can be efficiently reduced on a low dimensional basis. We present this application as an example of how suitably designed data-driven models can be combined with physical models to avoid relying entirely on data.

In a second study, we investigated the abilities of the CD-ROM memory architecture to extract relevant frequencies from system data and retain them in memory. We underline this aspect of the approach as a major advantage over less interpretable approaches such as the LSTM network. Indeed, this interpretability of the memory can be used to accurately initialize the model's memory (see Section 4.3.4) and confirm its validity.

Despite these advantages, we underline the fact that the CD-ROM approach is bound to linear dimensionality reduction methods as they are intrinsic to the Galerkin projection method. In the following Chapter, we discuss the fact that linear reduction is often limited for the identification of low-dimensional spaces for the dynamics of physical systems. On the other hand, we show that nonlinear methods can be combined with the theory of partially observed systems to identify very efficient, low-dimensional dynamical models from data.

CHAPTER 6

ILED: INTERPRETABLE LEARNING OF EFFECTIVE DYNAMICS FOR MULTISCALE SYSTEMS

Contents

6.1 Introduction 108

6.2 Methodology 109

6.2.1 Dimensionality Reduction 110

6.2.2 Framing *iLED* within the Mori-Zwanzig formalism . 111

6.2.3 The *iLED* architecture 113

6.2.4 Training the *iLED* architecure 115

6.3 Implementation Details 117

6.3.1 Memory Initialization 118

6.3.2 Linear Parameterization 118

6.3.3 Latent space centering 118

6.3.4 Form of the memory kernel network 119

6.4 Numerical Experiments 119

6.4.1 Example 1: The FitzHugh-Nagomo Model 119

6.4.2 Example 2: The Kuramoto-Sivashinsky Equation . . 122

6.4.3 Example 3: Navier-Stokes Equations for the Flow around a Cylinder 123

6.4.4 Remarks on the linearity of the dynamics 127

6.5 conclusion 128

6.1 Introduction

The two previous chapters introduced in detail the CD-ROM approach, which focuses on the improvement of the POD-Galerkin reduced order modeling method. We tried to show that the hybrid nature of the CD-ROM approach yielded multiple advantages in terms of training, stability and interpretability compared to purely data-driven approaches. Moreover, we discussed the similarities between *hybrid* modeling approaches and the topic of inductive bias in machine learning models, mentioned in Section 3.1.4.

In this chapter, we introduce the results presented in Menier et al. [192], where we apply similar ideas in the context of *nonlinear dimensionality reduction*. Indeed, it is a well-established fact that linear methods often fall short of optimality for the reduction of dynamical systems ([184]). By contrast, neural autoencoders introduced in Section (3.1.5) are able to efficiently capture the low dimensional, non-linear manifolds on which dynamical systems evolve ([135]) and have become state of the art for non-linear dimensionality reduction. The latent space identified by these autoencoders can then be used in combination with dynamical modeling approaches to construct reduced models of any system of interest.

We propose a novel interpretable model order reduction technique that leverages the efficiency of nonlinear dimensionality reduction for dynamical systems. This work is based on the existing *Learning Effective Dynamics* (*LED*) framework ([184]). The framework proposes to use neural networks to carry out dimensionality reduction to learn the structure of the latent space via a nonlinear mapping, and a second type of neural network architecture, the LSTM ([24], see also Section 3.2.1), to learn the potentially non-markovian dynamics of the reduced system. While this framework yields promising results, we show in this chapter that it can be modified to learn a theoretically grounded, interpretable dynamical model replacing the LSTM currently in use. To this end, we propose a novel machine-learning framework that is closely based on Mori-Zwanzig [6, 10] and Koopman-Operator theory ([1], see also Section 3.2.1). This theoretical basis yields a method that is both accurate and offers a higher degree of interpretability than classical deep-learning modeling approaches. The final model is constructed around interpretable linear dynamics and completed by a physically motivated nonlinear closure. Thus, we name the proposed framework *interpretable Learning of Effective Dynamics (iLED)*, a fully data-driven modeling approach that is well grounded in dynamical system theory. Once again to help frame the approach in the context of the thesis, the *iLED* method is placed on the *hybridization spectrum*, displayed in figure 6.1.

Due to a neural architecture closely based on Mori-Zwanzig and Koopman

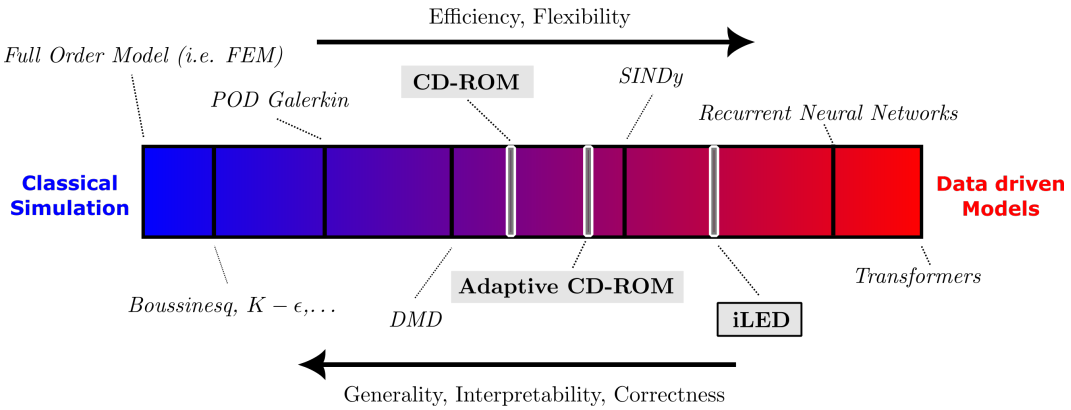


Figure 6.1: *iLED* on the hybridization spectrum. *NB: This figure is provided for illustrative purposes and helps frame our proposals within the broader context of the thesis. The placement of each method on the spectrum is open to discussion, and small variations could lead to displacing a given approach closer to one end or the other.*

theory our framework has a strong inductive bias and differs from black-box models such as Neural ODE and SDE [99, 136] which can be used to learn time continuous dynamical models. In fact, the *iLED* model is conceptually closer to the Koopman-based approaches discussed in Section 3.2.1 which only learn linear dynamics in the latent space of an autoencoder such as Otto and Rowley [90] and Champion, Brunton, and Kutz [114]. Indeed these models are very interpretable as linear dynamics are easy to analyze and simulate, but they assume that the model is Markovian, thus, they are unable to capture memory effects on the trajectory.

The remainder of this chapter is structured as follows. In Section 6.2 we present the general methodological framework giving special attention to the connection between our novel framework and the Mori-Zwanzig formalism as well as the Koopman operator theory. Computational aspects related to training the framework and generating predictions are discussed in Section 6.2.4. Numerical illustrations are then presented in Section 6.4. Finally Section 6.5 concludes the chapter with a discussion of the results and possible extensions.

6.2 Methodology

This Section introduces the novel *iLED* framework in detail. We especially focus on showing the connection between our method and theoretical considerations involving the Mori-Zwanzig formalism and the Koopman Operator theory. In Section 6.2.1, the motivation for the choice of neural networks to carry out dimensionality reduction is provided, while Section 6.2.2 presents the theoretical

justifications for our proposed interpretable reduced dynamics framework. Finally, the actual *iLED* architecture is summarised in Section 6.2.3.

As in the rest of the thesis, we consider as our target a high dimensional, potentially non-linear system whose state $\mathbf{u} \in \mathbb{R}^{d_{\boldsymbol{u}}}$ evolves in time according to an operator $\boldsymbol{F}$:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \boldsymbol{F}(t, \mathbf{u}) \tag{6.1}$$

This system can result from the discretization of a PDE such as the Navier-Stokes equations. Such high dimensional models are often derived from first principles and, while they are generally accurate, they can be extremely expensive to solve numerically.

The following of this Section is organized as follows. In Section 6.2.1, the motivation for the choice of neural networks to carry out dimensionality reduction is provided, while Section 6.2.2 presents the theoretical justifications for our proposed interpretable reduced dynamics framework. Finally, the actual *iLED* architecture is summarised in Section 6.2.3.

6.2.1 Dimensionality Reduction

As presented in chapter 2, the dimension $d_{\boldsymbol{u}}$ of the full order system in equation (6.1) can be exceedingly high compared to the actual intrinsic system dimension. Indeed, in general there exists a mapping $\mathcal{D} : \mathbb{R}^{d_{\boldsymbol{u}}} \mapsto \mathbb{R}^{d_z}$, with $d_z \ll d_{\boldsymbol{u}}$, such that $\mathbf{u} \approx \mathcal{D}(\boldsymbol{z})$.

In order to identify and exploit this reduced dimensionality, the mapping $\mathcal{D}$ can be extracted from data using machine learning methods. Reduced order modeling methods such as the POD-Galerkin approach which we considered in the earlier chapters of the thesis leverage linear reduction approaches to construct a basis that builds a matrix $\boldsymbol{V} \in \mathbb{R}^{d_{\boldsymbol{u}} \times d_z}$ on which both the system's state $\mathbf{u}$ and the dynamics $\boldsymbol{F}$ can be projected:

$$\boldsymbol{V}^T \mathbf{u} = \boldsymbol{z}, \tag{6.2}$$

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = \boldsymbol{V}^T \boldsymbol{F}(t, \boldsymbol{V}\boldsymbol{z}) + \boldsymbol{\epsilon}, \tag{6.3}$$

where $\boldsymbol{\epsilon}$ is an unknown error term. These linear reduction approaches have the important advantage of being physics-based as they are able to retain parts of the original model $\boldsymbol{F}$. Despite numerous successes with ROM[46, 186, 73], linear reduction has been shown to be inefficient, in terms of dimensionality reduction when compared to non-linear reduction approaches. Indeed, the

dynamics of most systems are not restricted to low dimensional linear subspaces but rather evolve on strongly non-linear manifolds[184]. This topic was discussed in detail in Section 2.2 where we showed that neural networks could be used to perform non-linear reduction efficiently. Thus, we propose to learn two parameterized non-linear mappings, a decoder $\mathcal{D}(\cdot;\boldsymbol{\theta}_{\mathcal{D}})$ and an encoder $\mathcal{E}(\cdot;\boldsymbol{\theta}_{\mathcal{E}})$, such that:

$$\mathbf{u} = \mathcal{D}(\boldsymbol{z};\boldsymbol{\theta}_{\mathcal{D}}), \tag{6.4}$$

$$\boldsymbol{z} = \mathcal{E}(\mathbf{u};\boldsymbol{\theta}_{\mathcal{E}}) \tag{6.5}$$

where $\boldsymbol{\theta}_{\mathcal{D}}$ and $\boldsymbol{\theta}_{\mathcal{E}}$ are the parameters of the decoder and encoder that are learned during training of the neural networks.
However, using a non-linear encoder/decoder structure, the dynamics of the reduced-order system have to be learned afterward or concurrently, as non-linear dimensionality reduction does not allow for the direct reduction of the original model $\boldsymbol{F}$. Existing works [132, 76], most notably the *LED* framework [184], have demonstrated that these reduced dynamics could be directly learned using recurrent neural networks:

$$\boldsymbol{z}_{t+1} = \mathrm{RNN}(\boldsymbol{z}_t, \boldsymbol{h}_t; \boldsymbol{\theta}_{RNN}), \tag{6.6}$$

where $\boldsymbol{h}$ is a memory term and $\boldsymbol{\theta}_{RNN}$ are the parameters of the RNN. At the same time, these models have limited interpretability, and aren't directly linked to dynamical systems theory. In the following Paragraphs, we show that it is possible to derive a interpretable reduced dynamical model, with theoretical justifications.

6.2.2 Framing *iLED* within the Mori-Zwanzig formalism

The *iLED* framework is based on both the Mori-Zwanzig formalism [6, 10] as well as Koopman operator theory [1, 55]. In this Section, we first give a reminder on the Koopman operator, introduced in Section 3.2.1, which acts on observable functions $\mathbf{g}$ of the state $\mathbf{u}$ of high-dimensional systems, before introducing the Generalized Langevin Equation (GLE) for a reduced subset of these observables. We subsequently define an appropriate closure term for the GLE and introduce a neural network architecture.

Koopman Operator Theory and the Generalized Langevin Equation

The Koopman operator, introduced in Section 3.2.1, describes the dynamics of observables of physical systems and has been employed extensively within reduced-order modeling approaches [153, 47, 55]. For the high-dimensional

system $\mathbf{u}(t, \mathbf{u}_0)$, a Koopman operator can be used to represent the dynamics of the system instead of Equation 6.1.

In more details, an observable $\mathrm{g} : \mathbb{R}^{d_{\boldsymbol{u}}} \mapsto \mathbb{R}$ of the system $\mathbf{u}$ is advanced in time by the Koopman operator $\mathcal{K}_t$ [161]:

$$\mathcal{K}_t \mathrm{g}(\mathbf{u_0}) = \mathrm{g}(\mathbf{u}(t, \mathbf{u_0})) \tag{6.7}$$

Operator $\mathcal{K}_t$ is linear, and potentially infinite dimensional. For practical purposes, its operating space can be separated in an observed subspace ($\mathcal{H}_{\mathbf{g}}$) defined as the space spanned by a chosen set of M observables $\mathcal{M} = \{\mathrm{g}_i\}_{i=1,\dots,M}$ and an orthogonal subspace $\mathcal{H}_{\bar{\mathbf{g}}}$ for which a set of basis functions $\overline{\mathcal{M}} = \{\bar{\mathrm{g}}_i\}_{i=M+1,\dots,\infty}$ can be constructed so that $\langle \mathrm{g}_i, \bar{\mathrm{g}}_j \rangle = 0$ for all $i \in [1, M], j > M$. The dynamics of observables can then be expressed on the basis defined by the set $\mathcal{M} \cup \overline{\mathcal{M}}$ ([161]):

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \mathbf{g}_{\mathcal{M}} \\ \mathbf{g}_{\overline{\mathcal{M}}} \end{bmatrix} = \mathbf{L} \begin{bmatrix} \mathbf{g}_{\mathcal{M}} \\ \mathbf{g}_{\overline{\mathcal{M}}} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{\mathcal{M}\mathcal{M}} & \mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}} \\ \mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}} & \mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}} \end{bmatrix} \begin{bmatrix} \mathbf{g}_{\mathcal{M}} \\ \mathbf{g}_{\overline{\mathcal{M}}} \end{bmatrix}. \tag{6.8}$$

where $\mathbf{L}$ is a linear operator that corresponds to the infinitesimal generator of $\mathcal{K}_t$, $\mathbf{g}_{\mathcal{M}} = [\mathrm{g}_1, \mathrm{g}_2, \dots, \mathrm{g}_M]$ are the chosen observables and $\mathbf{g}_{\overline{\mathcal{M}}} = [\bar{\mathrm{g}}_{M+1}, \bar{\mathrm{g}}_{M+2}, \dots, \bar{\mathrm{g}}_{\infty}]$ are the orthogonal observables. Note that the operator $\mathbf{L}$ is separated in four parts, with $\mathbf{L}_{\mathcal{M}\mathcal{M}}$ the dynamics in the observed subspace, $\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}$ the orthogonal dynamics and $\mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}}$ and $\mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}}$ the exchanges between the observed and orthogonal subspaces.

The above system can be solved for $\mathbf{g}_{\overline{\mathcal{M}}}$ as follows:

$$\mathbf{g}_{\overline{\mathcal{M}}}(t) = \int_0^t e^{(t-s)\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}} \mathbf{g}_{\mathcal{M}}(s) ds + e^{t\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{g}_{\overline{\mathcal{M}}}(0), \qquad t > 0. \tag{6.9}$$

Finally, by injecting Eq.(6.9) in (6.8), an expression for the dynamics of the observables $\mathbf{g}_{\mathcal{M}}$ is obtained:

$$\frac{\mathrm{d}\mathbf{g}_{\mathcal{M}}}{\mathrm{d}t} = \mathbf{L}_{\mathcal{M}\mathcal{M}} \mathbf{g}_{\mathcal{M}} + \mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}} \int_0^t e^{(t-s)\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}} \mathbf{g}_{\mathcal{M}}(s) ds + \mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}} e^{t\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{g}_{\overline{\mathcal{M}}}(0). \tag{6.10}$$

The above expression describes the dynamics of the partially observed state of a system and has the same form as the Generalized Langevin Equation derived in the Mori-Zwanzig formalism. It still depends on the unobserved part of the initial condition ($\mathbf{g}_{\overline{\mathcal{M}}}(0)$) via the last term and thus is not a closed equation for $\mathbf{g}_{\mathcal{M}}$ only. However, this last term is often modeled as noise or simply ignored in several modeling approaches [40, 44, 69, 161]. In the following Section the conditions under which this term can be accounted for are explained in more detail.

Closing the GLE

The last term in Eq.(6.10) depends on information that is unavailable as it is orthogonal to the observed subspace. However, this term vanishes if the history of the observed subspace is known, and the orthogonal dynamics ($\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}$) are dissipative. Indeed, if the history of the system is known, we can re-write Eq.(6.9) for any initial condition $[\mathbf{g}_{\mathcal{M}}(-\tau), \mathbf{g}_{\overline{\mathcal{M}}}(-\tau)], \tau > 0$:

$$\mathbf{g}_{\overline{\mathcal{M}}}(t) = \int_{-\tau}^{t} e^{(t-s)\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}} \mathbf{g}_{\mathcal{M}}(s) ds + e^{(t+\tau)\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{g}_{\overline{\mathcal{M}}}(-\tau). \tag{6.11}$$

The last term in Eq.6.11 vanishes for $\tau \to \infty$, if the orthogonal dynamics $\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}$ are dissipative, which is often a reasonable assumption as, for instance, when the orthogonal (unobserved) subspace corresponds to the small scales of a dynamical system. Under such hypothesis, we obtain the following closed equation for the dynamics of the observed subspace:

$$\mathbf{g}_{\overline{\mathcal{M}}}(t) = \int_{-\infty}^{t} e^{(t-s)\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}} \mathbf{g}_{\mathcal{M}}(s) ds, \tag{6.12}$$

$$\implies \frac{\mathrm{d}\mathbf{g}_{\mathcal{M}}}{\mathrm{d}t} = \mathbf{L}_{\mathcal{M}\mathcal{M}} \mathbf{g}_{\mathcal{M}} + \mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}} \int_{-\infty}^{t} e^{(t-s)\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}} \mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}} \mathbf{g}_{\mathcal{M}}(s) ds. \tag{6.13}$$

In the following subsection, we show that the various operators expressed in the closed GLE (Eq. (6.13)) can be learned from data to derive an interpretable and theoretically sound model for the reduced dynamics of physical systems.

6.2.3 The *iLED* architecture

To construct the *iLED* architecture, we first identify the observables $\mathbf{g}_{\mathcal{M}}$ with the learned subspace of the neural encoder $\mathcal{E}$ so that $\mathbf{g}_{\mathcal{M}} \equiv \boldsymbol{z} = \mathcal{E}(\mathbf{u}; \boldsymbol{\theta}_{\mathcal{E}})$. We then learn the various operators L_{xx} that express the different parts of the Mori-Zwanzig formalism in equation (6.13).

The observed dynamics $\mathbf{L}_{\mathcal{M}\mathcal{M}}$ can be directly learned as a linear operator, denoted $A_\theta \in \mathbb{R}^{d_z \times d_z}$ below. However, because both operators $\mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}}$ and $\mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}}$ are possibly infinite dimensional, they need to be approximated. We propose to learn these operators as non-linear transformations of the observables $\boldsymbol{z}$. Justifications for this choice will be detailed in Section 6.2.3.

We introduce two neural networks $\boldsymbol{\Psi}_1(\cdot;\theta) : \mathbb{R}^{d_h+d_z} \mapsto \mathbb{R}^{d_z}$ and $\boldsymbol{\Psi}_2(\cdot;\theta) : \mathbb{R}^{d_z} \mapsto \mathbb{R}^{d_h}$, where d_h is a user-defined parameter, and model the orthogonal dynamics $\mathbf{L}_{\overline{\mathcal{M}}\overline{\mathcal{M}}}$ as a negative diagonal operator $\Lambda_\theta \in \mathbb{R}_-^{d_h \times d_h}$. This choice is in line with the assumption that the orthogonal dynamics are dissipative and significantly simplifies certain computations such as the initialization of the non-markovian (or memory) term in the model. This leads to the *iLED* architecture in Figure 6.2.

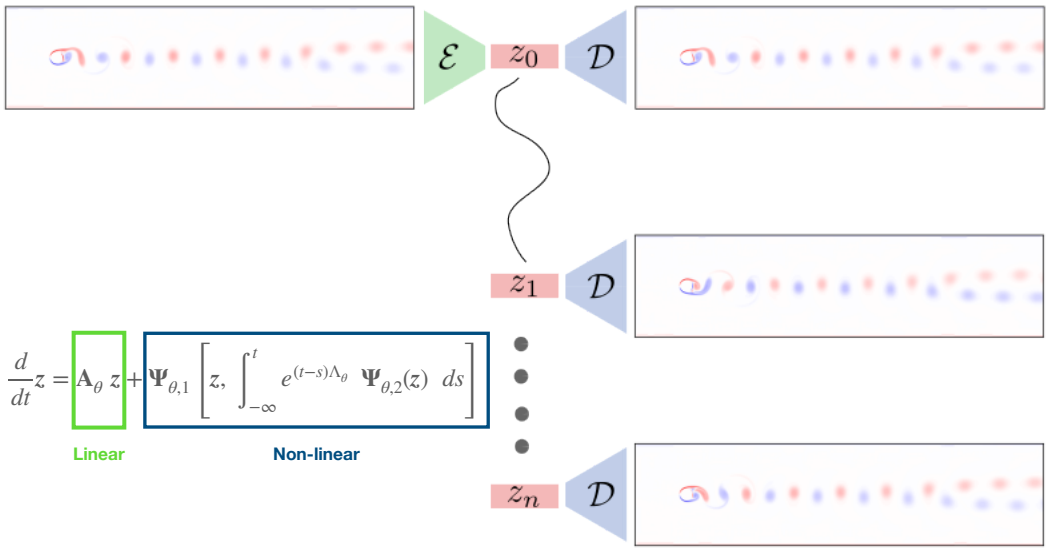


Figure 6.2: ILED architecture: The high-dimensional system is encoded to a lower-dimensional representation using the encoder $\mathcal{E}$. The lower-dimensional representation is propagated in time using a linear and a non-linear part based on the Mori-Zwanzig formalism. With the help of a decoder $\mathcal{D}$, the high-dimensional system is subsequently reconstructed.

The key part of this novel architure is the temporal dynamic of the *iLED* state $\boldsymbol{z}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z} = \mathbf{A}_\theta \boldsymbol{z} + \boldsymbol{\Psi}_{\theta,1}\left[\boldsymbol{z}, \int_{-\infty}^{t} e^{(t-s)\Lambda_\theta}\boldsymbol{\Psi}_{\theta,2}(\boldsymbol{z})ds\right] \tag{6.14}$$

We note that the operator Ψ_1 now also takes as argument the reduced state $\boldsymbol{z}$ itself. This choice simplifies the training of the model by putting at the beginning of the training process more weight on the latest time step during the computation of the memory. This remains coherent with the framework derived above, as the additional terms correspond to zero memory contributions which is equivalent to entries of the matrix Λ_θ going to negative infinity. This final *iLED* architecture allows us to directly learn the various terms of the Mori-Zwanzig formalism from data. Details on the training strategy will be given in Section 6.2.4. But we first provide additional justifications for the model.

Remarks on the Approximation of an infinite linear operator using a finite non-linear neural network based model

Our choice to approximate $\mathbf{L}_{\overline{\mathcal{M}}\mathcal{M}}$ and $\mathbf{L}_{\mathcal{M}\overline{\mathcal{M}}}$ with Deep Neural Networks is based on the universal approximation theorem. Neural networks are universal approximators for non-linear operators if both the given input and output of the operator are compact [22]. This has been successfully employed to construct Neural Operators such as the Deep Operator Network [163]. However, in our situation here, the subspace is infinite, and thus not compact. Fortunately, the Koopman operator, our starting point in Equation 6.7, is generally represented by a finite-dimensional operator with reasonable accuracy. In fact, this is a key assumption for all main data-driven Koopman models, see e.g. Li et al. [88], Otto and Rowley [90], and Brunton et al. [153] for reference. We assume that the same assumption holds here, and thus are dealing with a finite-dimensional orthogonal space whose operators can be approximated by neural networks.

6.2.4 Training the *iLED* architecure

This Section details the training of the *iLED* architecture, and the specific loss function used.

A key difficulty is the choice of the latent dimension d_z. The best choice is a dimension close to the intrinsic dimension of the problem at hand. In the common case where it is unknown, several approaches can be used to select this parameter. A first option is to directly apply hyperparameter optimization approaches such as grid search to the problem. *i.e.*, train autoencoders with increasing latent dimensions and select the dimension when the reconstruction performance of the autoencoder starts to plateau. However, this approach can

be expensive when applied to problems that use high-dimensional representations such as two-dimensional fluid flows. To avoid these expensive computations, statistical analysis such as correlation analysis can be employed to approximate the dimension of the attractor.

Having chosen this latent dimension d_z, we can set up the model for the latent dynamics.

To simulate the *iLED* model, we re-arrange the integro-differential equation (6.14) into a coupled system of ordinary differential equations. First, we define an intermediate term $\boldsymbol{h}$ as follows:

$$\boldsymbol{h}(t) = \int_{-\infty}^{t} e^{(t-s)\Lambda_\theta} \boldsymbol{\Psi}_{\theta,2}(\boldsymbol{z}) ds. \tag{6.15}$$

This $\boldsymbol{h}$ term corresponds to the *memory* of the model, which can be advanced in time in parallel of the reduced order state as follows:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} &= \mathbf{A}_\theta \boldsymbol{z} + \boldsymbol{\Psi}_{\theta,1}(\boldsymbol{z}, \boldsymbol{h}), \\ \frac{\mathrm{d}\boldsymbol{h}}{\mathrm{d}t} &= \boldsymbol{\Psi}_{\theta,2}(\boldsymbol{z}) + \Lambda_\theta \boldsymbol{h}. \end{aligned} \tag{6.16}$$

With this time-continuous architecture, the *iLED* model can be used in combination with any standard ODE integrator. In this work, we used the semi-implicit Runge-Kutta (siRK) scheme from Kar [39], which advances the *iLED* state $[\boldsymbol{z}, \boldsymbol{h}]$ in time as follows:

$$\begin{aligned} \left(I - \frac{k}{6}\Delta t \mathbf{A}_\theta\right) \boldsymbol{z}_{k\Delta t/3} &= \boldsymbol{z}_0 + \frac{k}{6}\Delta t \mathbf{A}_\theta \boldsymbol{z}_0 + \frac{k}{3}\Delta t \boldsymbol{\Psi}_{\theta,1}(\boldsymbol{z}_{(k-1)\Delta t/3}, \boldsymbol{h}_{(k-1)\Delta t/3}), \\ \left(I - \frac{k}{6}\Delta t \boldsymbol{\Lambda}_\theta\right) \boldsymbol{h}_{k\Delta t/3} &= \boldsymbol{h}_0 + \frac{k}{6}\Delta t \boldsymbol{\Lambda}_\theta \boldsymbol{h}_0 + \frac{k}{3}\Delta t \boldsymbol{\Psi}_{\theta,2}(\boldsymbol{z}_{(k-1)\Delta t/3}), \\ k &= 1, 2, 3. \end{aligned} \tag{6.17}$$

Where $\boldsymbol{\Lambda}_\theta$, $\boldsymbol{\Psi}_{\theta,1}$, $\boldsymbol{\Psi}_{\theta,2}$ and $\mathbf{A}_\theta$ correspond to the various operators introduced in Eq.(6.14). This scheme takes advantage of dynamics that efficiently separate a linear and a non-linear part. Moreover, siRK is very efficient for the simulation of stiff dynamics, which is critical here, as the *iLED* model can be stiff and unstable before being fully trained.

The siRK integration scheme is used in combination with the NeuralODE [99] method introduced in Section 3.2.3 to train the *iLED* architecture. We train the model in an *end-to-end* fashion, that is to say, both the neural autoencoder $\{\mathcal{E}, \mathcal{D}\}$ and the dynamics are optimized simultaneously, using the combined loss:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \alpha \mathcal{L}_{\text{forecast}}. \tag{6.18}$$

where $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{forecast}}$ are respectively the reconstruction and forecast losses, and α controls their relative importance.

The reconstruction loss $\mathcal{L}_{\text{rec}}$ drives the autoencoder to accurately reconstruct the true full order trajectory $\boldsymbol{u}^\star_{t_i}$:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \|\boldsymbol{u}^\star_{t_i} - \mathcal{D}(\mathcal{E}(\boldsymbol{u}^\star_{t_i}))\|_2^2. \tag{6.19}$$

The forecast loss $\mathcal{L}_{\text{forecast}}$ pushes the model to accurately predict the reduced state z:

$$\mathcal{L}_{\text{forecast}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \|\hat{\boldsymbol{z}}_{t_i} - \mathcal{E}(\boldsymbol{u}^\star_{t_i})\|_2^2, \tag{6.20}$$

where $\hat{z}$ is calculated according to Equation 6.16. This aggregated loss is sufficient to train the *iLED* architecture. However, we found that adding certain terms of lesser importance was beneficial and helped stabilize training. Thus, we added a reconstructed forecast loss:

$$\mathcal{L}_{\textit{rec forecast}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \|\boldsymbol{\Phi}^\star_{t_i} - \mathcal{D}(\hat{\boldsymbol{z}}_{t_i})\|_2^2. \tag{6.21}$$

And a regularization loss on the nonlinear part of the *iLED* dynamics:

$$\mathcal{L}_{\textit{non-linearity}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \|\boldsymbol{\Psi}_1(\hat{\boldsymbol{z}}_{t_i}, \hat{\boldsymbol{h}}_{t_i})\|_2^2, \tag{6.22}$$

where $\boldsymbol{\Psi}_1$ is the non linear part of the *iLED* dynamics in equation (6.14). Finally, the full loss is written as follows:

$$\mathcal{L} = \mathcal{L}_{\textit{rec}} + \alpha_1\mathcal{L}_{\textit{forecast}} + \alpha_2\mathcal{L}_{\textit{rec forecast}} + \alpha_3\mathcal{L}_{\textit{non-linearity}}, \tag{6.23}$$

with the coefficients α_i adjusted to control the importance of each term. In the next Section, we provide more details regarding lesser implementation details that were used to obtain the results presented in Section 6.4.

6.3 Implementation Details

This Section discusses the choices we made during the construction of the method that should be considered to reproduce the results.

6.3.1 Memory Initialization

Because the *iLED* and CD-ROM models use the same memory architecture, the memory is initialized in the exact same way:

$$h_0 = \int_{-\infty}^{0} \mathbf{\Psi}_1(\boldsymbol{z}(s)) e^{-\mathbf{\Lambda} s} ds. \tag{6.24}$$

As described in Section 4.3.4, the infinite boundary of the above integral is relaxed by computing the longest time horizon τ_{max} of the memory from the largest entry λ_{min} of the negative diagonal matrix $\mathbf{\Lambda}$:

$$\tau_{max} = \frac{\epsilon}{-\lambda_{min}} \tag{6.25}$$

Where $\epsilon \in \mathbb{R}^+$ is a small parameter, generally chosen to be equal to 10^{-2}, that controls the relative error on the computation of h_0. After relaxing the infinite boundary in equation (6.24), the memory can be initialised as follows:

$$h_0 = \int_{\tau_{max}}^{0} \mathbf{\Psi}_1(\mathcal{E}(\mathbf{\Phi}^\star(s))) e^{-\mathbf{\Lambda} s} ds. \tag{6.26}$$

Note that the above integral can be computed from the training data as a simple trapezoidal integration, which can be directly backpropagated through during training.

6.3.2 Linear Parameterization

To ensure a higher degree of stability in the model. The linear operator $\mathbf{A}_\theta$ in the *iLED* architecture is parameterized to be stable as follows:

$$\mathbf{A}_\theta = \mathbf{W}_\theta - \mathbf{W}_\theta^T - \mathrm{diag}(\mathrm{abs}(\vec{\boldsymbol{w}}_\theta)), \tag{6.27}$$

with $\mathbf{W} \in \mathbb{R}^{d_z \times d_z}$ a trainable weight matrix and $\vec{\boldsymbol{w}}_\theta \in \mathbb{R}^{d_z}$ a trainable vector. With this formulation, the operator $\mathbf{A}_\theta$ is guaranteed to be stable *i.e.* its eigenvalues have negative or zero real parts. This not only stabilizes the model but also avoids divergence of the model in the early stages of training.

6.3.3 Latent space centering

To allow for the interpretability of the linear term in the *iLED* dynamics, it is important to ensure that the latent codes computed by the encoder $\mathcal{E}$ are centered. Indeed, a limit cycle arising from an unforced linear system will necessarily be centered around the origin. To do so, we define a *LatentSpaceCentering* operation $\mathbf{LC}(z)$ as follows:

$$\mathbf{LC}(\boldsymbol{z}) = \boldsymbol{z} - \vec{\mu}. \tag{6.28}$$

Where $\vec{\mu}$ is a running mean of the latent code's averages that is computed during training and frozen at inference time. This approach is very similar to classical batch normalization, except the data is only centered, as unitary scaling of the latent space is not required for the model to learn efficiently.

6.3.4 Form of the memory kernel network

Once again, the memory architecture of the CD-ROM model is re-used, thus, the *lifting* operator $\mathbf{\Psi_2}$ is expressed as a concatenation of the state and trainable nonlinear transformations of said state $\boldsymbol{z}$. So as to exploit the information embedded in the latent space:

$$\mathbf{\Psi}_2(\boldsymbol{z}) = [\boldsymbol{z}, \mathcal{MLP}(\boldsymbol{z})]. \tag{6.29}$$

Where $\mathcal{MLP} : \mathbb{R}^{d_z} \mapsto \mathbb{R}^{d_h - d_z}$ denotes a standard multi layer perceptron.

6.4 Numerical Experiments

The capabilities of *iLED* are demonstrated on three relevant simulation problems: The FitzHugh-Nagomo model, a simple 1D equation with periodic dynamics; The chaotic dynamics presented by the Kuramoto-Shivasinsky equation; The incompressible Navier-Stokes equations describing flow around a cylinder with two different Reynolds numbers (100 and 750).

6.4.1 Example 1: The FitzHugh-Nagomo Model

The FitzHugh-Nagomo model was developed to study the dynamics of excitable systems. It has been widely used in biology, physics and neuroscience. The model consists of a pair of coupled Partial Differential Equations that describe the dynamics of a fast-acting variable $u(x,t) \in \mathbb{R}, x \in \Omega = [0, L], t \in [0, T]$, inhibited by a slower variable $v(x,t) \in \mathbb{R}$:

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + u - u^3 - v, \tag{6.30}$$

$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + \epsilon \left(u - \alpha_1 v - \alpha_0\right). \tag{6.31}$$

where variable u evolves on a much smaller time scale than its inhibitor v. This separation of time scales is controlled by parameter ϵ, set here to $\epsilon = 0.006$. The other model parameters are chosen as follows: $D_u = 1$, $D_v = 4$, $L = 20$, $\alpha_0 = -0.03$ and $\alpha_1 = 2$, to replicate the experiment presented in Vlachas et al. [184].

The computational domain Ω is discretized using a grid of $N = 101$ points. The problem is solved starting from 5 different initial conditions using the Latice-Boltzmann method [38] and its implementation provided in Vlachas et al. [184]. The data is sampled at rate $\Delta t = 1s$ to obtain 5 trajectories of 451 seconds each. Two of those trajectories are set aside for validation and the others are used for training. An additional trajectory of 10^4 seconds is simulated for testing purposes.

By training various autoencoders to reconstruct the training trajectories described above, we determined that the optimal latent dimension was $d_z = 2$, as the reconstruction accuracy evaluated from the validation trajectories saturates for higher dimensions. This result is coherent with the oscillatory nature of the dynamics and highlights the efficiency of non-linear dimensionality reduction. Indeed, a linear method such as PCA requires up to 16 latent dimension (see Vlachas et al. [184] *figure 2-A*) to achieve the same level of accuracy. A visualization of the system evolution, as well as the corresponding latent trajectory are presented in Figure 6.3.

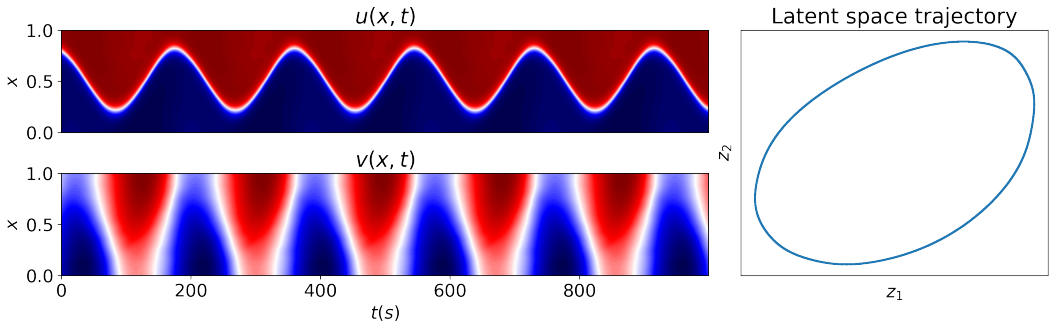


Figure 6.3: Visualization of the FHN model's dynamics. The evolution of the full state for a subset of the test trajectory is presented on the left. The right hand side of the plot displays the latent manifold learned by an autoencoder using latent dimension $d_z = 2$.

An *iLED* dynamical model is also trained at the same time as the autoencoder, using the procedure described in Section 6.2.4 (the hyperparameters used are detailed in 10.1.1). Figure 6.4 presents the results obtained by simulating the final model on the test trajectory. The Figure shows that the *iLED* model is able to accurately reconstruct the full order system state from the latent code z. Moreover, the dynamics is accurately captured: the model remains on the true latent attractor even after a very long integration.

Finally, we argue that the *iLED* method is particularly well-suited for this case, and highly interpretable. Due to the optimal latent dimension $d_z = 2$, the linear part of the *iLED* dynamics exhibits a single natural frequency, aligning with the periodic nature of the dynamics under study. The learned frequency is approximately $5.74mHz$, while the primary frequency extracted from the true system

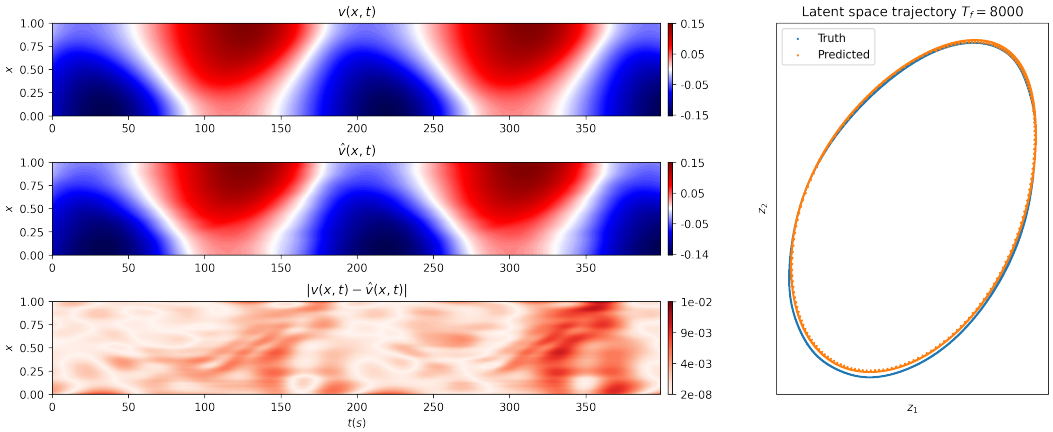


Figure 6.4: Forecasting performance of the *iLED* method on the FHN case. From top to bottom: true inhibitor field, predicted inhibitor field and absolute error between the two. The right hand side presents the true and predicted latent trajectory for an integration period of $8000s$. *NB:* only the inhibitor v field is presented for clarity, as it is harder to predict than the activator field u.

data using a Fourier Transform is $5.37mHz$. This comparison demonstrates that the operator has accurately captured the driving frequency of the system, allowing the linear part of the *iLED* model to support most of the dynamics. A close examination of the norm of the dynamics separately for the linear and non-linear terms (Figure 6.5) confirms this result: The figure clearly shows that the dynamics is mainly supported by the linear term, the contribution of the nonlinear term being approximately one order of magnitude smaller. It is important to note that the nonlinear term still plays a role in this case, as the learned latent attractor is not perfectly circular: A purely linear model would inevitably diverge from the true trajectory.

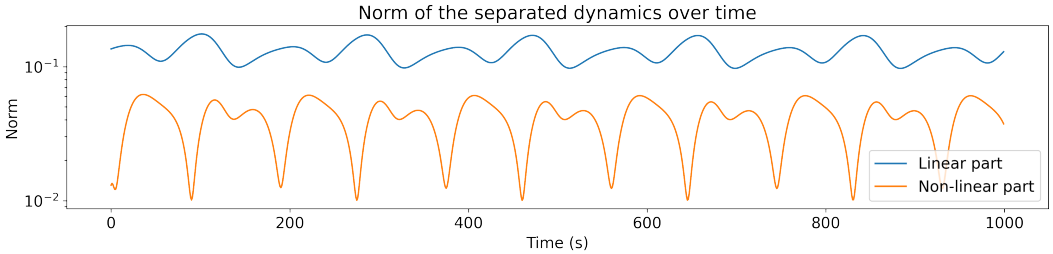


Figure 6.5: Norm of the dynamics parts

6.4.2 Example 2: The Kuramoto-Sivashinsky Equation

The Kuramoto-Sivashinsky (KS) equation, previously introduced in Section 4.5.3, provides a simplified mathematical description of the spatiotemporal dynamics of a range of physical systems, most notably in CFD. It serves as a prototypical example of a nonlinear partial differential equation, and exhibits a rich variety of behaviors, including the emergence of self-sustained oscillations, the formation of coherent structures, and the occurrence of spatiotemporal chaos, making it the perfect test-bed for reduced order modeling methods.

The KS equation can be written as:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} + u\frac{\partial u}{\partial x} = 0, \\ &u(x,t) \in \mathbb{R},\ x \in [0, L],\ t \in [0, T], \\ &u(0,t) = u(L,t), \end{aligned} \tag{6.32}$$

where u(x, t) represents the unknown scalar field, and L is the length of the computational domain, that controls the nature of the dynamics. Note that contrary to the CD-ROM experiments, we didn't target the problem of parametric modeling in this study, the viscosity ν in Eq.(4.29) is set to be equal to 1 and is thus discarded from the equations. We use here $L = 22$, a common value for the study of this problem ([184, 170]) which yields a dynamical system that evolves on a stable attractor with a characteristic dimension approximately equal to 8 (a higher dimensional attractor than the attractor of the FitzHugh-Nagomo model studied in previous Section). Moreover, the KS system develops chaotic dynamics under these conditions, which significantly increases the complexity of the learning problem, as small errors naturally compound over time during the simulation.

The problem is discretized on a spectral basis of N=64 Fourier modes, and advanced in time using a Semi implicit Runge-Kutta scheme [39]. We generate 2048 training trajectories starting from random initial conditions, and 64 others for validation. The initial conditions are all advanced in time for 3000 "warm-up" steps of length $\delta t = 0.025s$, which are discarded as they account for the transition from the random initial conditions to the chaotic attractor. The next 1280 steps are then sub-sampled with a $\Delta t = 0.25s$ in order to obtain the training and validation data. Finally, one hundred new initial conditions are simulated with a longer time horizon ($800s$) for testing purposes. The evolution of one of the training trajectories is presented in figure 6.6, as well as a visualization of the joint probability density $p(\frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2})$, which is a helpful way of visualizing the dynamics of the KS equation.

Applying the *iLED* method[1], we find that the reconstruction performance of the autoencoder used for dimensionality reduction does not improve for latent

[1]Details on the architecture and hyperparameters used can be found in 10.1.2

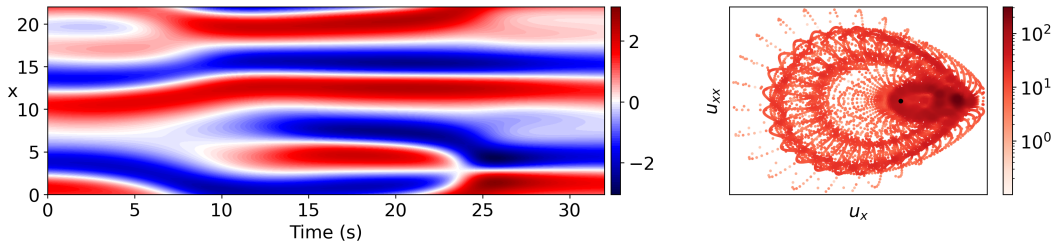


Figure 6.6: Two views of a training trajectory for the Kuramoto-Shivasinsky case (see text).

dimensions superior to $d_z = 8$, which is in accordance with the true intrinsic dimension of the KS attractor. Figure 6.7 presents results obtained on a test trajectory with a trained *iLED* model.

The figure clearly demonstrates that the *iLED* method is able to correctly capture the dynamics of the problem on a previously unseen trajectory, for a time horizon at least as long as its training horizon. Of course, the forecasting error does increase for longer integration times. But this was expected, as the chaotic nature of the problem makes it increasingly hard for a model to accurately follow the true system trajectory. Moreover, the figure shows that despite leaving the true trajectory, the obtained attractor, visualized through the densities of the derivatives, remains correct.

Similarly to the FHN case (section 6.4.1), let us take a close look at the eigenvalues of the learned linear operator in the *iLED* architecture. Figure 6.8 shows the natural frequencies learned by the *iLED* models after training under ten different random seeds. Although these learned frequencies are harder to interpret than for the FNH, as the KS system is not driven by a single main frequency, it is interesting to note that the different model initializations led to learning a similar range of frequencies. Moreover, the natural frequencies of the *iLED* linear operator are coherent with the frequencies observed in the data. Figure 6.8 displays the Fourier transform of a test trajectory, showing that a large range of frequencies is present in the data. The figure also shows that this range is covered by the various frequencies learned by the *iLED* linear operator, suggesting that while the chaotic attractor does not directly correspond to a periodic cycle in latent space, this cycle is still relevant to the system dynamics.

6.4.3 Example 3: Navier-Stokes Equations for the Flow around a Cylinder

Finally, we apply the *iLED* method to the case of the cylinder flow. The complexity of this case is controlled by the Reynolds number (*Re*), which is a dimensionless number that relates to the importance of energy dissipation in the system. We chose to tackle the simulation problem under two different Reynolds num-

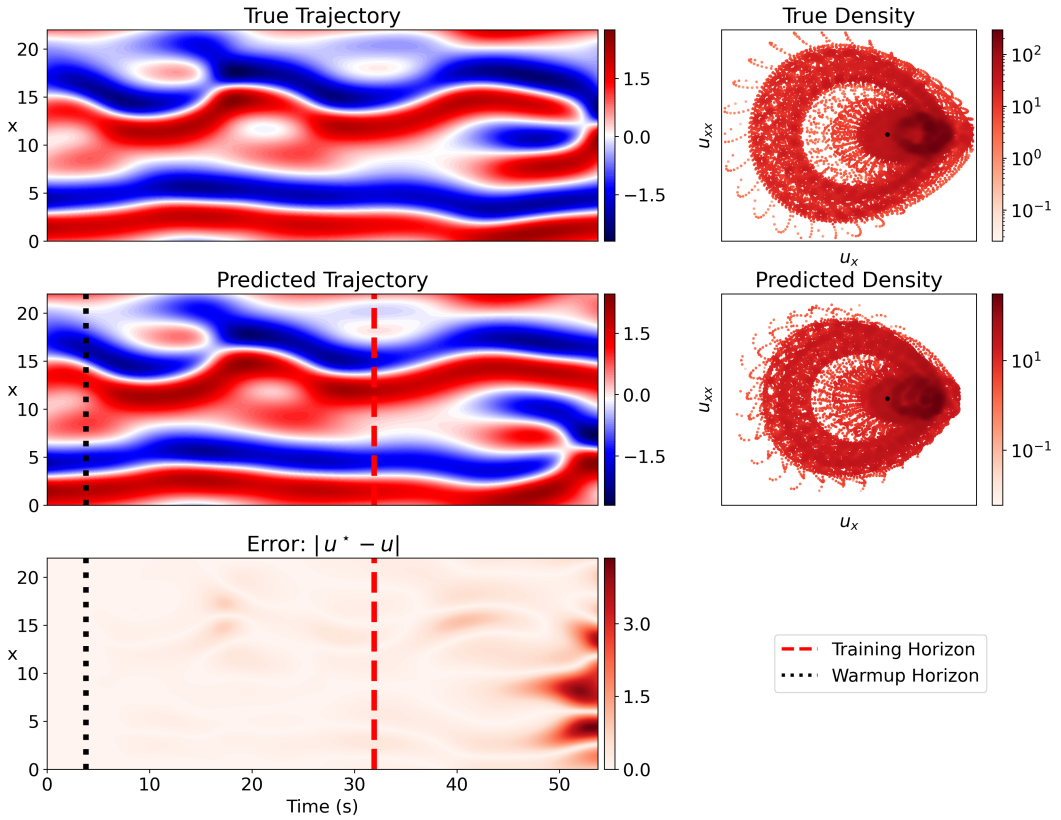


Figure 6.7: Results obtained with the *iLED* method on a test trajectory. *Dashed black line*: Horizon of the warm-up required to initialize the memory of the model, *Dashed red lined*: time horizon used to train the model.

bers: $Re = 100$, the standard value used to benchmark reduced order modeling applications; $Re = 750$, as the system then exhibits much more complex dynamics.

In both cases, the incompressible Navier-Stokes equations are solved using an adaptive meshing and time stepping solver [200]. The generated data is then interpolated on a cartesian grid to ensure compatibility with convolutional neural networks. To construct the autoencoder, we use a multiscale approach similar to the one proposed in Kičić et al. [189]. Indeed, the more complex part of the dynamics takes place around the cylinder, requiring a higher resolution than the remaining of the computational field. Thus, we use separate convolutional encoders with different resolutions for the domain around the cylinder and the remaining of the computational domain. These two encoders produce two intermediate latent codes z_1 and z_2 which are passed through an additional *mixer* multi layer perceptron to compute the latent code z. This *mixer* network is used to ensure that each dimension of the latent state z can encode information for both the higher and lower resolution parts of the state, which is important

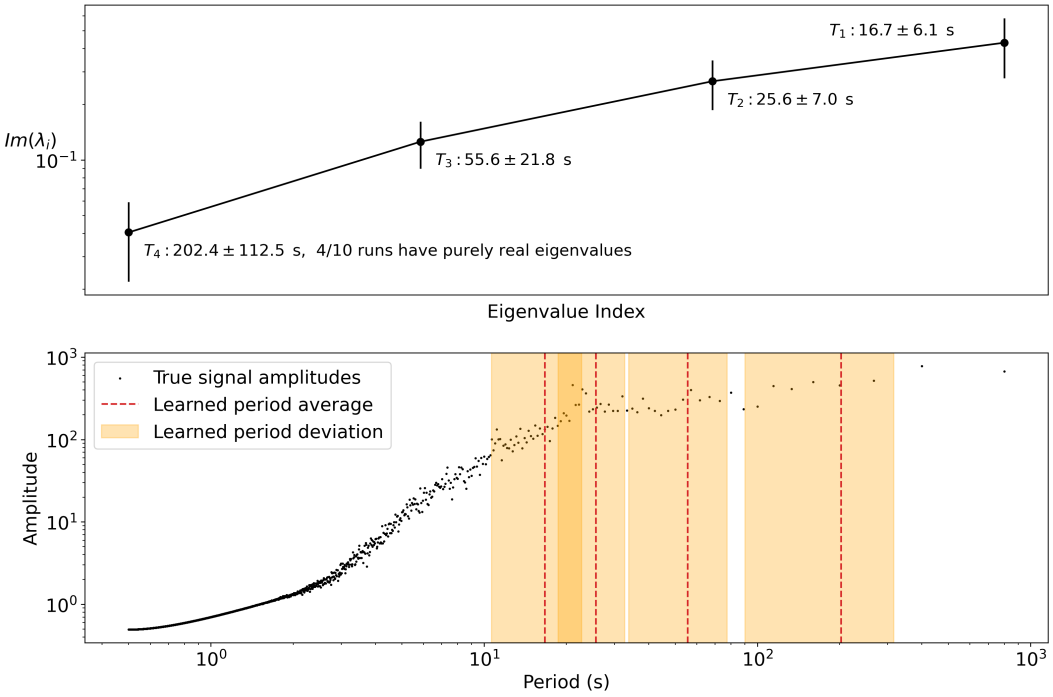


Figure 6.8: *Top*: Eigenvalues λ_i of the *iLED* linear operator (average and std. dev. over ten different training runs). *Bottom*: Fourier transform of a test trajectory averaged over the computational domain, the natural periods of the *iLED* operator are also displayed for comparison. Note that several runs learned one purely real eigenvalue, meaning that the learned period is infinite, thus not included in the computation of the largest period (T_4).

as the *iLED* linear operator $\mathbf{A}_\theta$ acts on the full latent state z. This multi-scale architecture is illustrated in figure 6.9. Additional details on the architecture and hyperparameters used can be found in 10.1.3

For both Reynolds numbers, the problem is simulated for $100s$. The first warm-up twenty seconds are discarded as they correspond to the transition from the initial condition. The rest of the trajectory is sub-sampled with a $\Delta_t = 0.02s$ yielding a trajectory of 4000 points. The first 2500 points are used for training, and the last 1500 are set aside for validation.

The results obtained by training an *iLED* model for the $Re = 100$ case are presented in figure 6.10. We used a latent dimension of $d_z = 3$, which is slightly higher than the minimal dimension 2 required to represent the limit cycle of the system, but yielded better modeling performance according to the combined loss (Eq.(6.18)). Figure 6.10 clearly shows that the *iLED* model is able to accurately reconstruct the system state after multiple periods of the dynamics. Similarly to the Fitz-Hugh Nagomo case (sec 6.4.1) the results underline the effectiveness of the *iLED* architecture, as the figure shows that most of the dynamics are sup-

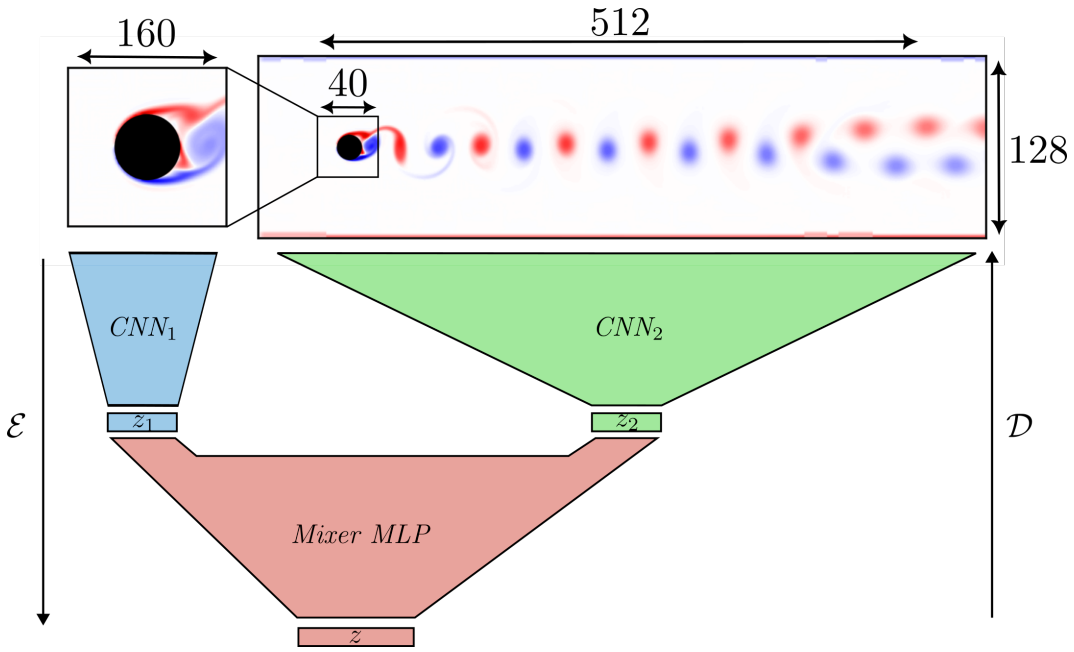


Figure 6.9: Multiscale architecture used to model the cylinder flow. The area around the cylinder is rendered at four times the resolution of the rest of the field, as it is where the dynamics are most complex.

ported by the linear part of the model.

Finally, the natural frequency of $1.466Hz$ learned by the *iLED* linear operator is in accordance with the system data which presents a dominant frequency of $1.562Hz$: This further confirms the validity of the model.

The results obtained on the case of the cylinder flow under a Reynolds number of 750 are presented in figure 6.11. Because this case presents more complex dynamics than the simple 2D periodic limit cycle encountered for $Re = 100$, we used a latent dimension of $d_z = 16$ to model the latent dynamics. This choice of latent dimensions was made in accordance with the results presented in Kičić et al. [189], because of the similarities with the *multiscale* autoencoder used in this work. Similar to the $Re = 100$ case, the *iLED* model is able to accurately forecast and reconstruct the system state and once again, despite the higher complexity of the case, most of the dynamics are supported by the linear operator and the neural network closure ($\mathbf{\Psi}_1$ in equation (6.16)) is only used to correct the numerical imperfections in the curvature of the learned latent attractor. Figure 6.11 also shows that the learned frequencies are coherent with the system data, as the two first natural frequencies of the linear operator are perfectly coherent with the dominant frequencies of the Fourier transform of the true latent trajectories.

With these results, we demonstrate the ability of the *iLED* model to scale to more complex, two-dimensional dynamics. The model yields satisfying perfor-

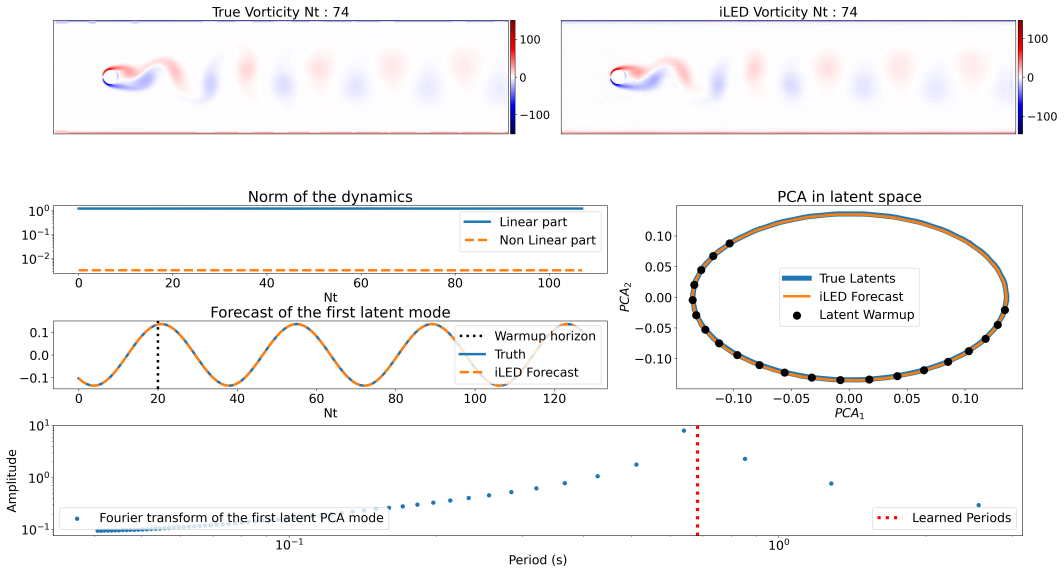


Figure 6.10: Results obtained with the *iLED* method on the case of the cylinder flow at a Reynolds number of 100.

mance combined with a high degree of interpretability and stability.

6.4.4 Remarks on the linearity of the dynamics

In two of the three numerical experiments presented above, we have shown that the *iLED* model was able to transform high dimensional, nonlinear PDEs into *quasi*-linear Ordinary Differential Equations. This is in fact coherent with the oscillatory nature of the dynamics in both the FHN and Cylinder flow cases. We underline this result as a major strength of the *iLED* framework as it is able to identify simple models from data without any *a-priori* assumptions on the system under study.

Moreover, it is important to note that although the identified models rely almost entirely on the linear part of the dynamics, they are still completed by a nonlinear term of lesser magnitude. Indeed, the complexity involved in learning a purely linear model reaching the same degree of accuracy might be higher. This is due to the fact that the neural autoencoders used for dimensionality reduction struggle to learn perfectly *organized* latent attractors, which is critical to ensure the accuracy of purely linear dynamics. Of course, this aspect is only magnified with the increasing complexity of the application case. Thus, the nonlinear term in the framework can be looked at as a relaxation of the constraints on the shape of the latent attractor while still allowing for the extraction of a simple interpretable model, as the observed low magnitude of the nonlinear dynamics allows for accurate analysis of the model from the learned linear term.

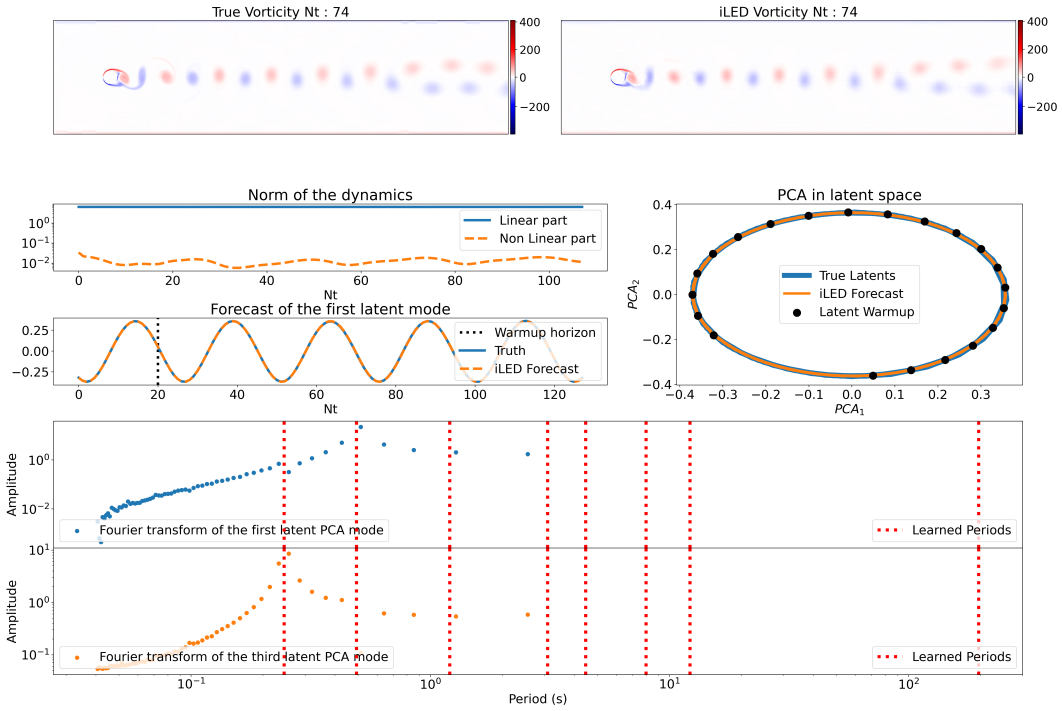


Figure 6.11: Results obtained with the iLED method on the case of the cylinder flow at a Reynolds number of 750.

Finally, we note that these *quasi*-linear dynamics were not observed in the case of the Kuramoto-Shivasinsky equations. This is due to the chaotic nature of the case. Because the system does not rely on a set of clearly identified driving frequencies, the linear part of the model is not sufficient for the accurate representation of the case, and the non-linear part then automatically learns to complete the dynamics. This once again underlines the *adaptability* of the model as no *a-priori* knowledge of the nature of the dynamics is required to model the system.

6.5 conclusion

This chapter introduced the *iLED* method proposed to learn interpretable reduced order dynamics for high-dimensional, multiscale systems. This method is closely based on Koopman operator theory and the Mori-Zwanzig formalism and thus benefits from the inductive bias derived from this domain knowledge. In addition to offering a high degree of interpretability, the latent dynamics of our novel approach are still expressive enough such that the method can be applied to various problems.

We show that the approach performs well on a range of dynamics, from chaotic problems to high-dimensional 2D flow cases. For each test case, the method is able to learn a linear model for the latent dynamics as well as a non-Markovian, non-linear closure term. The high-dimensional systems are mapped with a non-linear encoder to a latent space, in which the complex non-linear PDEs can be reduced to very simple *quasi*-linear ODEs, thus yielding fast and stable simulations. The high-dimensional state can be reconstructed from the latent space representation using a decoder that is trained simultaneously with the aforementioned encoder, using an autoencoder architecture.

Currently, the derived latent dynamics are deterministic. For future work, we plan to propose a probabilistic version based on either the Bayesian approach or Conformal Inference, in order to quantify the uncertainty caused by dimensionality and model reduction. Another unsolved challenge pertains to the optimal choice of the latent dimension. Indeed, we discussed in Section 6.2.4 the fact that various approaches can be used to estimate this value and that, although the most efficient, direct optimization of the latent dimension is not always feasible because of computational costs. Moreover, the *iLED* method could be used to model more complex problems such as partially observed systems, or applied to real-world problems with unknown dynamics such as epidemic dynamics or brain activity to help derive interpretable dynamical laws from available data.

Before concluding this thesis, the next chapter presents additional studies on topics adjacent to our work. We discuss possible ways to use results from dynamical systems theory to construct novel Deep Learning approaches, with a focus on generative modeling.

CHAPTER 7

ADDITIONAL STUDIES: DYNAMICAL SYSTEMS FOR GENERATIVE MODELING

Contents

7.1 Introduction 131
7.2 Unsupervised Domain Translation 132
7.2.1 Method 133
7.2.2 Results 135
7.2.3 Study Conclusion 137
7.3 Hamiltonian Flows for Generative Modeling . . . 137
7.3.1 Methods 139
7.3.2 Related Works 144
7.3.3 Experiments 146
7.3.4 Limitations and Open questions 150
7.3.5 Study Conclusion 152
7.4 Conclusion 152

7.1 Introduction

Aside from the main focus of the thesis, we carried out additional studies on the similarities between dynamical systems and Deep Learning, where instead of using deep learning to derive novel physical modeling approaches, we tried to use insights from dynamical systems theory to inform and construct novel neural architectures. The two studies presented in this Chapter focus on the use

of Hamiltonian Neural Networks [118], introduced in Chapter 3, to define invertible transformations. Section 7.2 first introduces work on the use of Hamiltonian Neural Networks for domain translation, which was presented in Menier et al. [177]. Section 7.3 then presents work on the use of Hamiltonian Neural Networks for generative modeling, where we show that despite limited performance, the Hamiltonian framework presents interesting properties for the construction of Normalizing Flows.

7.2 Unsupervised Domain Translation

Domain translation is the process of transforming elements from one domain to another. One can think of applications such as neural style transfer [80] which is for example used to apply a certain painter's style to photo-realistic images. A common problem encountered in domain translation applications is that, in many cases, paired data is not available during training, which means that the problem has to be formulated in an unsupervised setting. Unsupervised learning is very common in the field of generative modeling, and several architectures have been proposed to deal with the problem of Unsupervised Domain Translation. In this work, we focus on the Cycle-GAN [95] architecture, which has proved successful in various applications of Unsupervised Domain Translation [1].

Despite its success, the formulation of the Cycle-GAN method has been questioned and shown to be ill-posed. Using results from Chen and Gopinath [28], it can be shown that when considering two distinct domains, there exist an infinity of pairings between the two domains which satisfy the Cycle-GAN objective. This is an issue as the model could get stuck trying to learn wildly inefficient mappings, leading to unsatisfactory optima. This conditioning problem has been explored in depth by Bézenac, Ayed, and Gallinari [152], as they proposed to use a regularized residual network to learn the mapping between two given domains. Borrowing ideas from optimal transport and dynamical systems, they showed that pushing the training towards simple, low-energy, transformations in latent space leads to learning a sensible and trivially invertible mapping between the two domains of interest.

The study of the links between dynamical systems theory and deep learning is still to this day a major topic of interest. One can for example cite the identification of residual networks as first order approximations of a time-continuous process which has led to the development of ground-breaking approaches such as neural ordinary differential equations (Neural ODE [99]) or invertible neural networks [113].

Building on this existing connection, as well as the work of Bézenac, Ayed,

[1] Cycle GAN project page, https://junyanz.github.io/CycleGAN/

and Gallinari [152], we propose a formulation of unsupervised domain translation as a continuous time process with conservation guarantees which ensure invertibility by construction. The proposed architecture learns the dynamics of the transformation as a Hamiltonian dynamical system. Hamiltonian systems are typically used in General Mechanics to describe the evolution of conservative systems. They preserve a quantity, called the Hamiltonian, along their trajectory. Using neural networks to learn Hamiltonian dynamics is an earlier idea that was proposed in Greydanus, Dzamba, and Yosinski [118]. However this work proposes to use them to ensure invertibility of the generative process which is a desirable property to ensure the domain translation problem is well-posed. Learning conservative transformations is in fact critical to other generative modeling approaches, such as normalizing flows [74].

7.2.1 Method

Invertibility and CycleGAN

Formally, we can look at the two domains as two separate sets $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^d$, where d is the dimension of the space, *i.e.* the pixel space for images, or any latent representation space. The goal of unsupervised domain translation is to learn the forward mapping $F : \mathcal{A} \to \mathcal{B}$ as well as the reverse map $R : \mathcal{B} \to \mathcal{A}$ so that the pair (F, R) generates semantically meaningful samples of each domain. That is to say, the generated samples should be indistinguishable from samples in the target domain, while remaining coherent with their corresponding sample in the original domain.

CycleGan proposes to enforce these constraints by using a combined loss: $\mathcal{L} = \mathcal{L}_{adv} + \mathcal{L}_{cyc}$. The first term $\mathcal{L}_{adv} = \mathcal{D}(F(\mathcal{A}), \mathcal{B})$ corresponds to an adversarial loss which measures the distance between the generated samples and the target domain. This term ensures that generated samples are indistinguishable from the target domain. In CycleGAN, $\mathcal{D}$ is implemented using Generative Adversarial Networks [60].

The second term in the loss is called the cyclic loss, $\mathcal{L}_{cyc} = \|F \circ R(x_{\mathcal{A}}) - x_{\mathcal{A}}\| + \|R \circ F(x_{\mathcal{B}}) - x_{\mathcal{B}}\|$. This term promotes transformations F that are invertible and such that $R = F^{-1}$. Intuitively, this pushes the CycleGAN architecture towards learning minimal transformations of the samples, so as to retain a maximum of information from the initial sample and simplify the reconstruction $R \circ F$. This second term is used to ensure coherence between the translated and initial samples. In addition, learning an invertible (thus bijective) map between the two domains is critical at the conceptual level. Indeed, one sample from a given domain should not map to multiple samples in the target domain as only one sample in the target domain should optimally satisfy the trade-off between coherence with the original sample and similarity with the target domain.

Continuous models and Hamiltonian Neural Networks

The previous Paragraph outlined the importance of ensuring the translation map is invertible to relax the learning problem. In fact, this is not specific to the domain translation problem, as invertibility of learned maps has been linked to classical deep learning problems such as vanishing/exploding gradients in recurrent neural networks [51], or the training of other generative models like normalizing flows [74]. Several approaches have been proposed to push learned models towards invertibility [108, 2], however, they often impose significant constraints on the structure and expressivity of the models, leading to important training costs.

In this work, we propose to use a natural formulation for invertible transformations. Exploiting the parallel between the residual networks used in numerous image processing approaches, and ordinary differential equations, we propose to define domain translation as a continuous system. Starting at $t = 0$ with samples from one domain $x_{t=0} \in \mathcal{A}$, we learn a transport flow f_θ so that, at $t = T$, $x_{t=T} \in \mathcal{B}$:

$$\frac{dx}{dt} = f_\theta(x), \quad \text{s.t.} \quad x_0 \in \mathcal{A},\ x_T \in \mathcal{B} \tag{7.1}$$

This formulation is not enough to ensure invertibility of the transformation, as the flow f_θ could be dissipative, or even unstable. To enforce invertibility, we express the flow f_θ as a conservative operator using Hamiltonian neural networks inspired from Greydanus, Dzamba, and Yosinski [118]. To do so, the samples are divided into two vectors of equal length $x = [p, q]$, (we assume d to be even as a modeling choice). In general mechanics, p and q would respectively describe the position and momentum of the studied entities. In our setting, their significance is more abstract and is defined by another function, called the Hamiltonian $\mathcal{H}_\theta(p, q) : \mathbb{R}^{d/2} \times \mathbb{R}^{d/2} \to \mathbb{R}$, which we parameterize using a neural network, hence:

$$f_\theta(x) = \begin{pmatrix} \frac{dp}{dt} & = & -\frac{\partial \mathcal{H}_\theta}{\partial q} \\ \frac{dq}{dt} & = & \frac{\partial \mathcal{H}_\theta}{\partial p} \end{pmatrix} \tag{7.2}$$

Using Neural ODEs and automatic differentiation, the function $\mathcal{H}_\theta$ can be trained to satisfy the transport objective, *i.e.* $x_T \in \mathcal{B}$ given $x_0 \in \mathcal{A}$. Moreover, this formulation is invertible by design as it preserves the quantity $\mathcal{H}_\theta(x)$ along its trajectory. We show below that learning the transformation f_θ with this formulation allows for the generation of semantically correct samples, without using the cyclic loss required in CycleGAN. Thanks to the conservation properties of the flow f_θ, the inverse map is trivially obtained by integrating the flow backward in time:

$$x_{\mathcal{B}} = x_{\mathcal{A}} + \int_0^T f_\theta(x)dt \quad \Longleftrightarrow \quad x_{\mathcal{A}} = x_{\mathcal{B}} + \int_T^0 f_\theta(x)dt \tag{7.3}$$

7.2.2 Results

Generative results

We apply our Hamiltonian domain translation approach to image generation tasks. As proposed in Bézenac, Ayed, and Gallinari [152], we train an encoder E and a decoder G to map images from both domains to a latent space of size $d = 128$. This is a common approach in many image processing approaches, as the intrinsic dimension of a given image dataset is generally much lower than the pixel representation, which is the same idea as the one presented in Section 2.2 for dynamical systems. Thus, encoding images to a low-dimensional latent space reduces the domain translation problem complexity, as well as training costs.

Once the pair (E, G) is trained, it can be used to generate low-dimensional encoded vectors of images of the dataset at hand. We then use our approach to learn the transport flow f_θ. The Hamiltonian $\mathcal{H}$ and discriminator $\mathcal{D}$ are implemented as multi layer perceptrons with 3 hidden layers. The continuous flow is learned using the *optimise-then-discretise* version of NeuralODEs. We apply the architecture to the task of translating male samples of the celebA [70] dataset to females. Figure 7.1 presents samples generated with this approach.

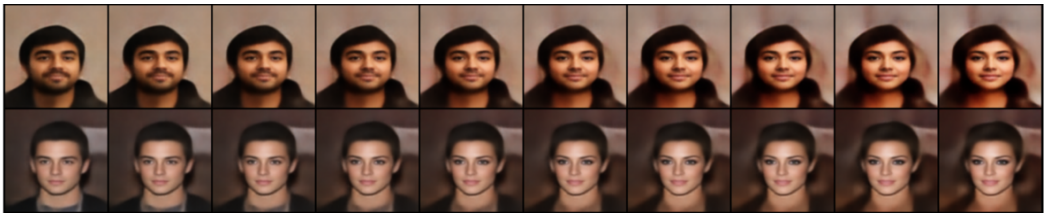


Figure 7.1: Selected samples of the male to female transport process using the proposed continuous domain translation approach. The transported encodings are decoded at regular time intervals, to illustrate the transformation applied by the model.

As shown on Figure 7.1, decoding the transported samples along the transformation trajectory shows that the flow f_θ progressively transforms the male samples to females. As expected, the conservative nature of the model promotes transformations that retain non gender-specific features, as we observe that attributes such as pose, skin tone, face shape and background are preserved during the transformation. Figure 7.2 demonstrates an additional interest of the Hamiltonian architecture as we are able to generate males from female samples by simply integrating the flow backward (see Eq. 7.3). One should

Figure 7.2: Selected samples of the reverse male to female generation process. NB : These samples are generated using equation (7.3) as transport flow was solely trained to map male to females.

note that these results were obtained without ever training the model to map females to males as we do not compute the cyclic loss used in CycleGAN. These results are similar to the results of Bézenac, Ayed, and Gallinari [152] while no penalization of the magnitude of the flow applied by the model is used but invertibility is enforced instead.

Figure 7.3: Results of excessive integration of the transport flow. A model trained to map males to females in 1 $t.u.$ is integrated backward for more than twice the map horizon. We observe that the generated samples retain semantic sense for up to about 1.5 times the training horizon.

Excessive Integration

An interesting feature of using a continuous flow to carry out domain translation is that one can gain some insight in the way the model transforms samples. If a flow f_θ has been trained to map two domains in one time unit (*t.u.*), $T = 1$, it can be integrated for a longer period, pushing the transformation further. This is one of the major differences between learning continuous transformations and discrete residual blocks. While residual blocks approximate the flow in specific regions of the latent space, the continuous flow is defined over the whole space. Any trained model starts losing performance once it drifts too far from its training conditions but we observed interesting results when integrating our model for several *t.u.*. Figure 7.3 shows that transported samples retain semantic meaning for up to about one and a half *t.u.*, as the model progressively adds more and more gender-related features such as beards, wider jaws, shorter hair, etc. This generalisation performance can be linked to the conservative architecture of the model which prevents it from diverging to un-

known conditions. It also adds to the interest of the approach as it supports the idea that the model is consistent with the structure of the latent space.

Training

It should be noted that, once the autoencoder is trained, learning the flow f_θ is very inexpensive. The model starts generating semantically coherent samples after a single epoch, and does not require fine tuning between the training of the discriminator $\mathcal{D}$ and the flow f_θ. More formal benchmarking against comparable domain translation methods are planned for the future.

7.2.3 Study Conclusion

This work proposes a novel formulation for domain translation. By using a time-continuous approach, we are able to leverage results from general mechanics to obtain a model that is invertible by construction. We show that this model can quickly learn to map two domains of interest, even in a latent space learned prior to training the domain translation architecture. We frame this study in the context of hybridization, showing that as Deep Learning can be applied to improve numerical modeling, the reverse also stands, and results from dynamical systems theory can be exploited to derive new Deep Learning approaches. The next Section follows the same line, where we discuss the way Hamiltonian flows can be used for generative modeling.

7.3 Hamiltonian Flows for Generative Modeling

Generative modeling is a prominent topic in Deep Learning research. The idea of being able to learn and sample from the data distribution is appealing in many applications. In recent years, denoising diffusion probabilistic models [75] have generated a lot of interest owing to their ability to generate samples from very complex image distributions [167, 182, 183]. Before these results, several works on generative models had already been proposed. One can cite: Variational Autoencoders (VAEs)[53], Generative Adversarial Networks (GANs) [61] and flow-based models [58]. Each method has its advantages and limitations. VAEs introduce a prior Gaussian distribution on the low dimensional latent representation of the data. They are easy to train but the assumption of Gaussian distribution can lead to *posterior collapse* due to uninformative latent representations or due to a high expressivity of the decoder [126, 125]. Remarkable achievements were obtained with GANs in image, text and music generation [120, 134, 101]. In GANs, two neural networks are trained in an adversarial

way to map a low dimensional latent Gaussian distribution to the data distribution. However, the Nash equilibrium between the training of the generator network and the discriminator network is difficult to achieve [130]. This makes GANs hard to train and prone to *mode collapse*. Due to the low dimensional latent representation of the data used to generate samples, the probability density estimation of the inputs cannot be estimated with GANs. Flow-based generative models overcome these limitations by progressively transforming a simple distribution into a complex data distribution through a sequence of invertible functions. Moreover, densities are preserved through the transformation.

These models were proposed in the early days of deep generative modeling, and are generally less efficient than competing approaches in terms of generated samples quality. The invertibility constraint introduces limitations on the choice of possible architectures. However, they have several advantages compared to alternative approaches:

- **Density Estimation**: flow-based models can be used to carry out density estimation, which is extremely useful for critical tasks such as molecular folding, or for predicting the probability of future events.
- **Meaningful Training Objective**: They can be trained using a likelihood-based objective which is quantitatively significant. This is not the case of approaches such as GANs.
- **Latent Representation**: They learn a well organised latent representation of the data at hand. Indeed manipulations of the normalized data can be carried out to achieve useful transformations in the latent space of flow-based generative models. The latent code of samples can for example be interpolated, or transported along chosen directions to obtain meaningful transformations of the data such as adding a smile to a face image, or smoothly transform a sample.

These desirable aspects make flow-based generative models an attractive option for certain generative modeling tasks, which is why a large body of recent work was dedicated to their development [85, 103, 119]. Their main limitation is their computational cost, as their training requires repetitive and costly evaluations of the determinant of a large jacobian matrix. This is usually addressed by using network architectures specifically designed to alleviate the cost of evaluating the training objective (more details on these architectures are given in Section 7.3.1 and in [58]).

Significant efforts have been made to relax these architectural constraints and lighten the computational burden. In this work, we propose to focus on the Continuous Normalizing Flow (CNF) approach, first introduced in parallel to the Neural ODE method [99], which models the flow as a time continuous transport of the data samples. The dynamics of this continuous transport map can

be learned by optimising a simplified maximum likelihood objective, without additional constraints on the architecture describing the flow.

In this Section, we leverage the link between continuous neural approaches such as the CNF method and well established results from general mechanics. the goal is to propose a variant of the Continuous Normalizing Flow approach, termed *Hamiltonian Normalizing Flow*. We show that the link between deep learning and dynamical systems can be exploited to ensure desirable characteristics such as invertibility by construction and simplify both the computation of the training objective and density estimation. Furthermore, we discuss the link between our proposal and classical flow-based modeling architectures.

The study is organized as follows. Section 7.3.1 introduces the theory behind flow-based generative modeling as well as our proposal for the improvement of Continuous Normalizing FLows. Section 7.3.2 discusses the links between our approach and existing works while results are presented in Section 7.3.3.

7.3.1 Methods

Maximum Likelihood training

The goal of flow-based generative modeling is to learn a model p_θ of an arbitrary data distribution. This is done by defining a continuous and invertible map F_θ trained to transform the data distribution $x \sim p^\star$ into a simpler latent distribution $z \sim p_G$. Such a transformation is usually implemented using neural networks, and trained by minimising the negative log-likelihood of the data, resulting in the following optimization problem:

$$\theta \in \arg\min_{\widetilde{\theta}} \mathop{\mathbb{E}}_{x \sim p^\star} \left[-\log p_{\widetilde{\theta}}(x) \right]. \tag{7.4}$$

Typically, the distribution p_G is defined as a normal distribution so that $p_G = \mathcal{N}(\mu_G, \text{diag}(\Sigma_G^2))$, and the negative log-likelihood objective can be computed using the following change of variable formula:

$$\begin{aligned} \log p_\theta(x) &= \log p_G(z) + \log \left| \frac{\partial F_\theta}{\partial x} \right|, \\ &\text{with } z = F_\theta(x). \end{aligned} \tag{7.5}$$

Thus, the optimisation of a flow-based generative model requires the estimation of the determinant of the jacobian of the model $\left| \frac{\partial F_\theta}{\partial x} \right|$ at each training iteration. This can be computationally expensive, which is why the form of the transformation F_θ must be carefully chosen. Usually, the transformation is implemented as a sequence of N intermediate steps $f_{\theta,n}$:

$$\begin{aligned} F_\theta &= f_{\theta,1} \circ f_{\theta,2} \circ \ldots \circ f_{\theta,N}, \\ \log \left| \frac{\partial F_\theta}{\partial x} \right| &= \sum_{n=1}^{N} \log \left| \frac{\partial f_{\theta,n}}{\partial x} \right|, \end{aligned} \tag{7.6}$$

where each step $f_{\theta,n}$ is a neural network specifically designed to be invertible. Indeed, neural networks are not natively invertible, nor is the determinant of their Jacobian matrix generally straightforward to compute. To address this issue, specific architectures called coupling layers are used [58, 85] to construct the intermediate transformation steps. These layers are invertible by construction and have a triangular jacobian matrix, greatly simplifying the computation of the second term in Eq. (7.5), since the determinant of their Jacobian is equal to the product of its diagonal terms so that:

$$\log \left| \frac{\partial f_{\theta,i}}{\partial x} \right| = \text{sum} \left(\log \text{diag} \left(\frac{\partial f_{\theta,i}}{\partial x} \right) \right). \tag{7.7}$$

Coupling layers make the training of flow-based generative models computationally tractable at the cost of constraints on the architecture of the networks used in the learning process. Alternative approaches rely on more flexible parameterizations of the transformation F_θ. In this work, we focus on the Continuous Normalizing Flows [99] approach which defines the map F_θ as a continuous transformation such that:

$$F_\theta(x_0) = x_0 + \int_0^T f_\theta(x_t) \mathrm{d}t, \tag{7.8}$$

where f_θ can be any function of the samples $x_0 \sim p^\star$ and $x_t \equiv x(t)$, $x(0) = x_0$. This allows for a more flexible parametrization of the function F_θ and changes the negative log-likelihood objective Eq. (7.5) as follows:

$$\log p_\theta(x_0) = \log p_G(x_T) + \int_0^T \mathrm{Tr} \left(\frac{\mathrm{d} f_\theta}{\mathrm{d} x_t} \right) \mathrm{d}t, \tag{7.9}$$

which is inexpensive to evaluate since a trace operation scales only linearly with the number of hidden dimensions, while the computation of the determinant in Eq. (7.5) scales with the cube of the dimension for dense jacobian matrices.

Despite these simplifications, Continuous Normalizing Flows are only invertible in the continuous time limit, and still require the evaluation of the network's jacobian at every integration step. In the following Section, we introduce an alternative perspective on learning invertible transformations using conservative dynamical systems. We also describe the way these transformations can be used to learn flow-based generative models.

Hamiltonian flows

Continuous transformations and Invertibility

As presented in the previous Section, the construction of flow-based generative models hinges on the invertibility of the transformation F_θ. This Paragraph

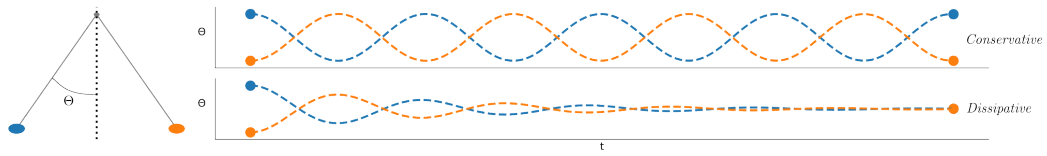


Figure 7.4: Illustration of the invertibility of conservative systems. The ideal (conservative) pendulum conserves its energy along the trajectory, thus, its initial condition can be retrieved from any point of the trajectory. The real (dissipative) pendulum loses energy over time, and converges to 0 for $t \to \infty$, preventing inversion.

presents how results from general mechanics can be used to inform the construction of Continuous Normalizing Flows to ensure their invertibility. Indeed, the properties of the transformation in Eq. (7.8) are directly related to the form of the transport dynamics f_θ.

Importantly, the invertibility of the transformation is not numerically guaranteed if the dynamics f_θ are unconstrained. This is especially the case if the dynamics f_θ have dissipative properties, as dissipative systems can forget their initial condition and yield an unstable inverse process (see figure 7.4 for an illustration). On the other hand, a conservative system defines an invertible transformation, as any point of the trajectory can be used to retrieve the initial condition. To ensure that f_θ is conservative, we suggest to model the function f_θ as a governing equation of a Hamiltonian system:

$$f_\theta(x = [q, p]) = \begin{pmatrix} -\frac{\partial \mathcal{H}_\theta([q,p])}{\partial p} \\ \frac{\partial \mathcal{H}_\theta([q,p])}{\partial q} \end{pmatrix}, \tag{7.10}$$

where the samples $x \in \mathbb{R}^{2d}$ are split into two equal parts $q, p \in \mathbb{R}^d$ and $\mathcal{H}_\theta$ is a scalar trainable function called the Hamiltonian. This idea was proposed in Greydanus, Dzamba, and Yosinski [118] to learn the dynamics of conservative systems. Indeed, the system in Eq. (7.10) is conservative by construction, meaning that no matter the structure of the function $\mathcal{H}_\theta$, the value of $\mathcal{H}_\theta(x)$ is conserved along the trajectory of the system. This formulation can thus be used to ensure the invertibility of Continuous Normalizing Flows by construction.

Volume preservation

A significant aspect of this choice of parameterization is that it is volume preserving, meaning that its divergence, here given by the trace of its jacobian, is null. This greatly simplifies the computation of the training objective Eq. (7.9) as the second term vanishes:

$$\log p_\theta(x_0) = \log p_G(x_T) + \int_0^T \cancelto{0}{\operatorname{Tr}\left(\frac{\mathrm{d}f_\theta}{\mathrm{d}x_t}\right)} \mathrm{d}t. \tag{7.11}$$

This result implies that conservative flows preserve probability densities along their trajectory. It is worth to note that the second term in the right hand side of Eq. (7.5) and Eq. (7.9) rescales the target distribution such that volumes are preserved when local compressions or stretchings of the probability measure arise under the map F_θ or the flow f_θ. Volume preservation can also be too constraining as the model is not allowed to compress nor stretch locally the probability measure. To overcome this limitation, we propose to augment the data with additional dimensions. This effectively allows the model to *stack* samples along the additional dimensions to increase or remove mass on the distribution marginalized to the original dimensions. This is illustrated in figure 7.5 where we augment a 1D bimodal distribution $q \sim p^\star$ with a 1-D Gaussian dimension $p_0 \sim \mathcal{N}(0, 1)$.

With this formulation, our Hamiltonian Normalizing Flow (HNF) can be trained to model compressive cases. Moreover, it can be used to easily carry out density estimation. Using the volume preserving properties of the Hamiltonian flow and the fact that the variables p_0 and q_0 are independent, we write:

$$p(x_0) = p(q_0)p(p_0) = p_G(x_T), \tag{7.12}$$

$$\implies p(q_0) = \frac{p_G(x_T)}{p(p_0)}. \tag{7.13}$$

Thus, the probability density of a data sample q_0 can be easily computed as the ratio of two Gaussian probability densities, without the need to evaluate the determinant of a jacobian matrix. We note that the accuracy of this estimator is conditioned by the performance of the model, *i.e.*, the accordance between the target distribution p_G and the true transported data distribution $x_T \sim \widetilde{p}_G$. Because models verify their training objective under some convergence precision, the estimator in Eq. (7.13) turns out to be noisy. This directly impacts the performance of the model in estimating the probability density. More details are provided in the results Section 7.3.3.

Training a HNF

The Hamiltonian Normalizing Flow can be trained by directly optimising the negative log-likelihood objective in Eq. (7.9). The augmented data samples x_0 are transported by integrating the Hamiltonian system Eq. (7.10) using a Runge-Kutta 4(5) time marching scheme and backpropagation is carried out using a *discretise-then-optimise* version of the NeuralODE approach [99, 175].

Note that the negative log-likelihood in Eq. (7.11) should be modified to reflect the augmentation of the data. Indeed, the model should minimize the probability of the data distribution $q_0 \sim p^\star$, not the augmented data distribution x_0. This can be easily computed using the density estimator (7.13):

$$-\log p_\theta(q_0) = -\log p_G(x_T) + \log p(p_0) \tag{7.14}$$

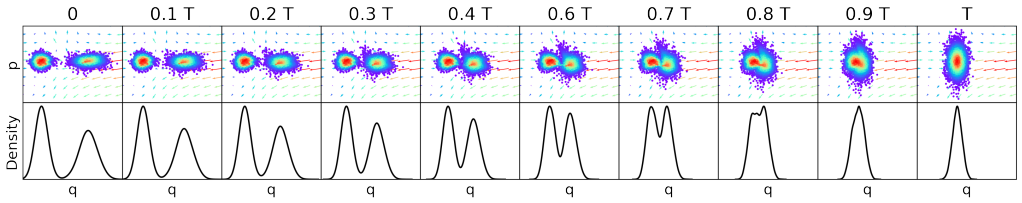


Figure 7.5: Gaussianization of a 1D bimodal distribution $q_0 \sim p^\star$ augmented with a Gaussian dimension $p_0 \sim \mathcal{N}(0, 1)$. Points are colored according to their probability in the final Gaussian $(q_T, p_T) \sim p_G$ distribution, as probabilities are conserved in the (p,q) space by the Hamiltonian Normalizing Flow.

Because p_0 follows a simple Gaussian distribution, this does not significantly increase the computation time of the training objective.

Relation to coupling layers

To improve understanding of the proposed method we make the link between the coupling layer architecture and Hamiltonian dynamics clearer. We show that, in certain cases, Hamiltonian systems and coupling layers are equivalent. We focus on a specific architecture called the Additive Coupling Layer which is the basis of modern flow-based generative modeling owing to its invertibility and volume preserving properties. Additive Coupling Layers advance a sample x_0 through the flow using the following rule:

$$\begin{aligned} \begin{bmatrix} q_0 \\ p_0 \end{bmatrix} &= x_0, \\ p_{\frac{1}{2}} &= p_0 + f_\theta(q_0), \\ x_{\frac{1}{2}} &= \text{permute}([q_0, p_{\frac{1}{2}}]). \end{aligned} \tag{7.15}$$

Where the *permute* operation is predefined exchange of the dimensions so that successive coupling layers act on different parts of the state. In contrast, we consider the integration of a Hamiltonian system using a leap-frog time marching scheme, specifically designed for the simulation of Hamiltonian dynamics. This time marching scheme advances the state $x_0 = (q_0, p_0)$ as follows:

$$\begin{aligned} p_{\frac{1}{2}} &= p_0 - \frac{1}{2} \frac{\partial \mathcal{H}_\theta}{\partial q}\bigg|_{q_0, p_{\frac{1}{2}}} \\ q_1 &= q_0 + \frac{1}{2} \left(\frac{\partial \mathcal{H}_\theta}{\partial p}\bigg|_{q_0, p_{\frac{1}{2}}} + \frac{\partial \mathcal{H}_\theta}{\partial p}\bigg|_{q_1, p_{\frac{1}{2}}} \right) \\ p_1 &= p_{\frac{1}{2}} - \frac{1}{2} \frac{\partial \mathcal{H}_\theta}{\partial q}\bigg|_{q_1, p_{\frac{1}{2}}} \end{aligned} \tag{7.16}$$

When the Hamiltonian is expressed as the sum of potential and kinetic energies ($\mathcal{H}_\theta([q,p]) = T_\theta(q) + V_\theta(p)$), and the permutation performed in Eq. (7.15) is a simple swap of the dimensions ($permute([q_0, p_{\frac{1}{2}}]) = [p_{\frac{1}{2}}, q_0]$), the leap-frog time stepping scheme becomes equivalent to a stack of three additive coupling layers:

$$\begin{aligned} p_{\frac{1}{2}} &= p_0 - \frac{1}{2}\frac{\partial V_\theta}{\partial q}\bigg|_{q_0} \quad \equiv \; p_{\frac{1}{2}} = p_0 + f_\theta(q_0) \\ q_1 &= q_0 + \frac{\partial T_\theta}{\partial p}\bigg|_{p_{\frac{1}{2}}} \\ p_1 &= p_{\frac{1}{2}} - \frac{1}{2}\frac{\partial V_\theta}{\partial q}\bigg|_{q_1} \end{aligned} \tag{7.17}$$

Although this equivalence does not hold in the general case as more complex coupling layer architectures are used in modern flow based models, this parallel can be used to further justify the choice of the Hamiltonian structure to build continuous flow-based generative models.

It should be noted that this separation of the Hamiltonian is not required in our approach and, in fact, our application focuses on the general case where $\frac{\partial^2 \mathcal{H}_\theta}{\partial p \partial q} \neq 0$. The trainable Hamiltonian function considers the full state, in contrast with classical discrete approaches, as coupling layers only consider part of the state at each flow step.

7.3.2 Related Works

Discrete Flows: As presented in the above Sections, state of the art flow-based generative models use the coupling layer architecture introduced in Dinh, Krueger, and Bengio [58] to build efficient and invertible flow models. These layers have been used and improved in various works; Dinh, Sohl-Dickstein, and Bengio [85] used an affine version of the coupling layer to allow the model to compress distributions and Kingma and Dhariwal [103] proposed to use invertible 1×1 convolution to increase the performance of the model. Other modifications of this architecture have been proposed over the years (see Ho et al. [119] for example), and overall, coupling layers have proven their efficiency for both density estimation and generative tasks. However, they impose a significant architectural constraint on the model. We propose our HNF as an alternative to coupling layers for the design of flow-based models, and in a more general sense, the design of invertible neural transformations.

Although **discrete models currently outperform our Hamiltonian Normalizing Flows on image datasets**, we emphasize the theoretical advantages of the Hamiltonian Normalizing Flow framework, as the proposed approach sig-

nificantly relaxes the constraints on the form of neural networks used in the model, simplifies the computation of the training objective and reduces the estimation of probability densities to a simple ratio of Gaussian probabilities.

Continuous Normalizing Flows : CNFs [99] have lower complexity than discrete models and have proven their efficiency. However, the cost of integrating the trace of their jacobian remains a bottleneck, although it can be efficiently approximated as shown in Grathwohl et al. [117]. The advantage of the HNF method is that this computational burden disappears as this specific term vanishes with our formulation.

Conservative Neural Applications: In the previous Sections, we described the link between the conditioning of the neural network describing the dynamics (f_θ) and the invertibility of the transformation. This topic has been studied in other works as it pertains to various application fields. Approaches such as Behrmann, Duvenaud, and Jacobsen [113] have proposed to penalise the conditioning of weight matrices to ensure asymptotic stability of residual networks while Haber and Ruthotto [86] have proposed to constrain weight matrices with a symplectic structure to ensure the stability and invertibility of the networks. Finally, Richter-Powell, Lipman, and Chen [181] have proposed a new parameterization for divergence free field, which could be used to obtain similar theoretical results as our method.

The Hamiltonian framework: The Hamiltonian framework appears in several works on generative modeling. Dockhorn, Vahdat, and Kreis [172] use the same idea of dimensionality augmentation as we do to allow for faster mixing of the noise with the data samples in diffusion models, yielding a system of ODE that can be decomposed in a Hamiltonian part combined with a Ornstein-Uhlenbeck process. Similarly, Huang, Dinh, and Courville [133] show that their augmented normalizing flows can be considered to be a discrete approximation of a Hamiltonian system. Most notably, the idea of using Hamiltonians systems to carry out flow-based modeling was also proposed in Toth et al. [146]. The authors propose to use a Hamiltonian system to forecast the state of a system by doing rollout generation of the next states. They also show that the Hamiltonian framework can be used to do pure generation on toy datasets. We consider the present work to be an extension of this work, with the following differences:

- **Continuity**: To train their flow model on toy datasets, the authors of Toth et al. [146] only use two leapfrog integration steps, making their approach a variation of an additive coupling layer flow (see Section 7.3.1). On the other hand, we show that the Hamiltonian framework can be used directly in combination with the NeuralODE approach, without giving consideration to the number of flow steps.

- **Hamiltonian architecture**: Previous works propose to express the Hamiltonian function as a linear combination of potential and kinetic energy,

$\mathcal{H}_\theta = T_\theta(q) + V_\theta(p)$. We show that this constraint is not necessary, allowing for a more flexible parameterization of the model.

- **Image generation**: We apply the Hamiltonian framework to image generation tasks and convolutional Hamiltonian functions.

7.3.3 Experiments

This Section presents the various experiments carried out to demonstrate the performance of Hamiltonian Normalizing Flows. We first discuss the case of a 1D bimodal distribution to illustrate several aspects of the model, particularly the density estimation process and its relation to the random dimensions used to augment the data. Examples of applications on 2-D toy datasets are then presented and finally, we demonstrate the ability of the Hamiltonian Normalizing Flow to scale to real datasets.

In all experiments, we used the adaptive checkpointing adjoint method [150] which is a more stable version of the Neural ODE method, but has higher memory requirements. As discussed in the above Sections, the Runge-Kutta 4(5) adaptive time stepping scheme is used to allow the model to learn the number of integration steps required to carry out the transformation. This can be thought of as adjusting the depth of the model during training. In the particular case of Hamiltonian systems, symplectic integration schemes would be favored for their volume preserving properties, however, these come with increased computational costs.

1D density estimation

To illustrate the behavior of the density estimator in equation Eq. (7.13), we use the case of the normalization of a 1D bimodal distribution denoted $p^\star$. As presented on figure 7.5, a Hamiltonian Normalizing Flow can be trained to learn the map between the augmented data distribution $x = [q_0, p_0], q_0 \sim p^\star, p_0 \sim \mathcal{N}(0, 1)$ and a Gaussian distribution $x_T \sim p_G$. In this 1D case, the Hamiltonian $\mathcal{H}_\theta$ is parameterised as a simple multi layer perceptron with three hidden layer of 64 neurons each and the swish activation function is chosen over the rectified linear unit function because it is continuously differentiable. Once the model is trained, the density probability of the data samples can be estimated using the formula Eq. (7.13) as illustrated in Figure 7.6. As mentionned in Section 7.3.1, the estimator is noisy because of the distance between the target distribution p_g and the true distribution generated by the model $x_T \sim \widetilde{p}_G$. However, the estimator is unbiased since it does yield the correct density value on average, this can be shown by sampling several p_0 values for each data sample q_0.

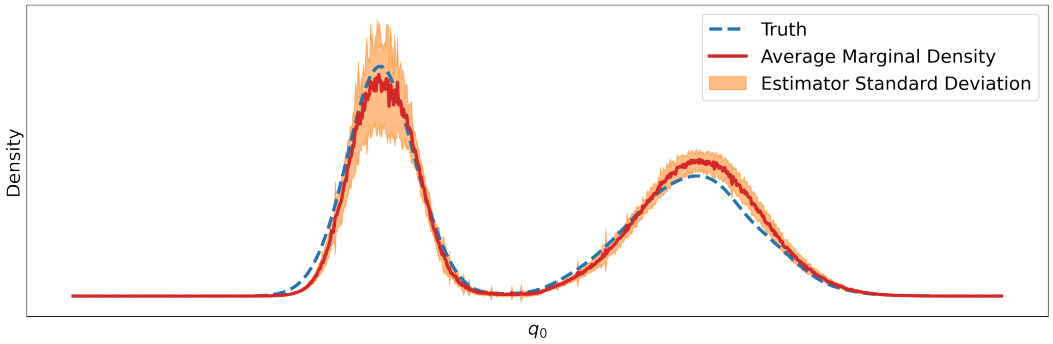


Figure 7.6: Density estimation carried out the estimator using the HNF. As discussed in sec 7.3.1, the estimator doesn't yield the same value depdending on the p_0 value sampled.

Toy Datasets

Following the results obtained on a one dimensional case, we apply *Hamiltonian Normalizing Flows* on several 2D toy datasets. For each dataset, the 2D data $q_0 \in \mathbb{R}^2$ is augmented with two 1D Gaussian dimensions $p_0 \in \mathbb{R}^2$. Similar to the experiment on the 1D distribution, the Hamiltonian function is parameterized as a multi layer perceptron with 3 hidden layers of 256 neurons each and SiLU activation functions.

Figure 7.7 presents the results obtained after training the models on the various cases. The figure shows that the model is able to accurately capture the target distributions, both in cases where distributions are disjoint and multimodal. It is interesting to note that the number of function evaluations required to carry out the normalization is of the order of 10, which is relatively low when compared to state of the art CNF results on similar benchmarks. For example, the improved CNF method proposed in Grathwohl et al. [117], which yields excellent results on both tabular and image data, requires about 100 function evaluations to learn similar toy datasets.

Real Data

Finally, this subsection presents results obtained by applying the Hamiltonian Normalizing Flow approach to real datasets. We first present an application to several tabular datasets which can be used to measure the performance of a likelihood model. Application of the HNF method on image datasets are then presented.

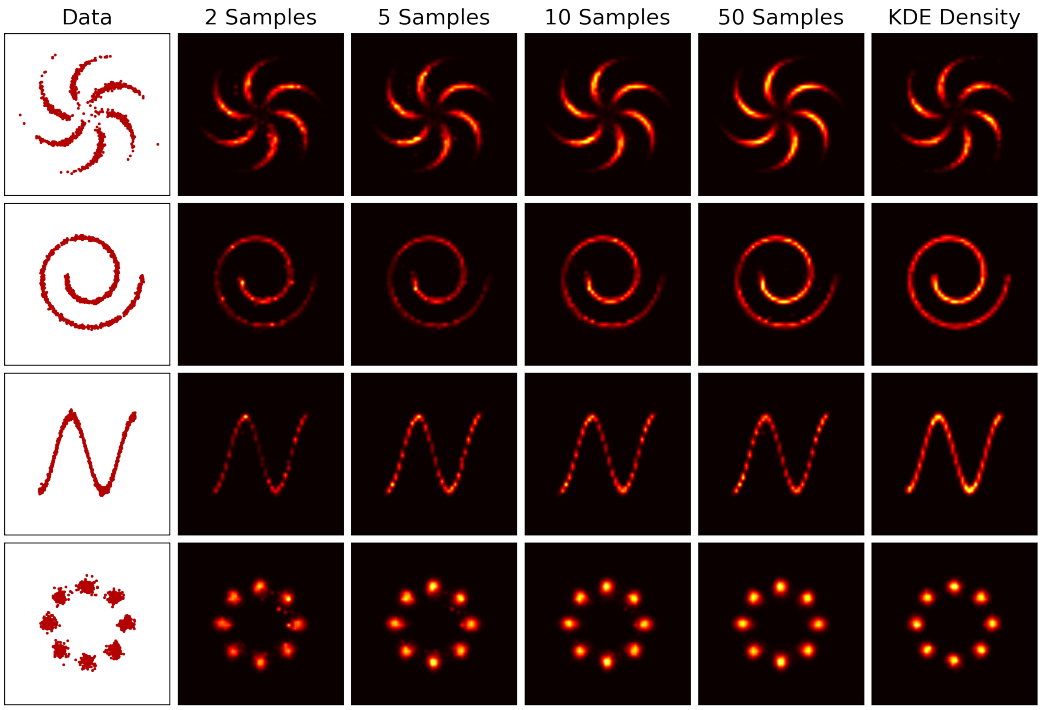


Figure 7.7: *Left column:* samples from the data distribution. *Center columns:* 2D toy densities averaged over different numbers of p_0 samples . *Right column:* Kernel density estimation.

Tabular Data

This Paragraph presents the application of the HNF method to likelihood estimation tasks on datasets commonly used to validate the performance likelihood models. These datasets are pulled from the UCI machine learning repository [54], and we reproduce the data pre-processing steps of Papamakarios, Pavlakou, and Murray [91] to allow for direct comparisons with other state of the art models. The hamiltonian is defined as a multi layer perceptron, and trained using the simplified negative likelihood objective Eq. (7.14).

Table 7.1: Average test compression costs in nats (lower is better) obtained on tabular datasets with the HNF method. Compared with results from Grathwohl et al. [117] and Papamakarios, Pavlakou, and Murray [91].

	GAS	HEPMASS	MINIBOONE
Real NVP	-8.3	18.71	13.55
Glow	-8.15	18.92	11.35
FFJORD	-8.59	14.92	10.43
HNF	-7.96	20.66	17

Figure 7.8: Generation of black & white faces at temperature of 0.7. The noise on the left is continuously transformed during to obtain the final samples.

Results on the selected datasets are presented in table 7.1. Although the HNF method does not surpass the performance of the well optimized state of the art flow-based modeling architectures, the performance of our proposed method is of the same order. This indicates that with a more thorough design of experiment and optimization of the training of the model, the method could perform on par with its state of the art counterparts, while offering a more flexible parameterization, and being less expensive to optimize.

MNIST

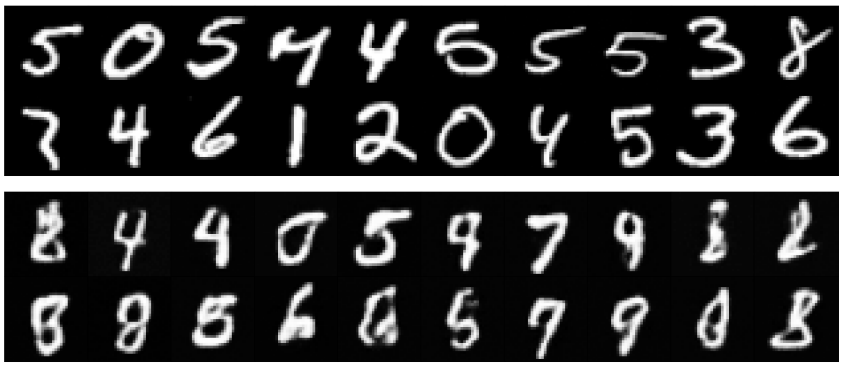


Figure 7.9: *Top*: Samples from the MNIST digits dataset. *Bottom*: Random samples generated with Hamiltonian Normalizing Flow at temperature 0.7

In this Paragraph, we present results obtained on image generation tasks. First, we focus on the MNIST handwritten digits dataset [198] which contain black and white 28 by 28 images. To build the Hamiltonian Normalizing Flow, we use the multiscale architecture proposed in [85], which consists of a sequence of L levels each containing K flows, followed by a squeeze operation that decreases the dimension of the images and increases their number of channel, and another K flow steps. In our application, we use $L = 2$ levels with $K = 2$ HNF on each side of the squeezing operation. The flow steps are expressed as

a Hamiltonian continuous transport, and the Hamiltonian at each flow step is a convolutional neural network that maps the current image to a scalar.

Results of the generation process on the MNIST dataset are presented on figure 7.9. Despite their imperfections, these results demonstrate the ability of the Hamiltonian Normalizing Flow method to scale to image datasets. They also show that the Hamiltonian formulation can be used in combination with arbitrary architectures, such as multiscale approaches and convolutional neural networks which leverage spatial correlations in the data.

Black and White Celebs

Finally, in an effort to scale the approach beyond 28 by 28 pixel images, we applied the approach on the normalization of a grayscale version of the celebA dataset [199] which contains two hundred thousands images of celebrities. After taking a center crop of the images, and resizing them to a size of 64 by 64 pixels, we trained a model to normalize the data using a sequence of 10 HNF, with the Hamiltonian parameterized as a convolutional neural network. This particular choice over the multiscale architecture used for the MNIST dataset is made to allow for the visualization of the continuous evolution of the samples during the generation process, as shown on figure 7.8, while samples from the model are presented on figure 7.10. These results demonstrate the potential of the method to scale to higher dimensional problems.

Figure 7.10: Random samples generated with a convolutional HNF at temperature 0.7

7.3.4 Limitations and Open questions

The previous Paragraphs have shown that the Hamiltonian Normalizing Flows framework presents interesting properties for the modeling of arbitrary data distributions. However, we want to underline that **there are still unanswered**

questions that should be addressed to make the method competitive with state of the art approaches. We identify two main issues below:

- **Density Estimation:** In the results, we have discussed the *noisy* nature of the density estimator provided by the Hamiltonian Normalizing Flow. We showed that the empirical results did not respect the fact that the density of a data point q_0 predicted by our model should be independent of the augmented variables p_0 used to carry out the normalization.

  We explain this issue by the inability of the model to converge to the optimum of the learning problem, *i.e.*, the normalized data distribution does not perfectly correspond to the target Gaussian distribution p_G. Thus, the accordance between the density $p_G(q_T, p_T)$ and the actual probability of the data varies for different p_0 values. A similar problem happens when carrying out data generation, where the model does not yield perfect independence between the generated data $\widetilde{q_0}$ and generated augmented variable $\widetilde{p_0}$.

  One idea to address this issue would be to carry out *bi-directionnal* training, by training at the same time the normalization process using the MLE objective of equation 7.14 and the generation process, where the distance between the true and generated data distributions could be estimated using Maximum Mean Discrepancy as in Ardizzone et al. [112].

  The possibility of leveraging this noisy nature of the density estimator to quantify model uncertainty should also be investigated.

- **Expressivity:** The generality of the approach and the expressivity of the network architecture require additional investigation. We have shown in Section 7.3.1 that the Hamiltonian normalizing flow could be seen a more general formulation of a sequence of additive coupling layers [58]. However, this is not sufficient to state that the Hamiltonian formulation is general, as better-performing approaches such as Kingma and Dhariwal [103] are now using more complex invertible layer architectures, which are harder to relate to our HNF.

  Another justification uses Liouville's theorem which can be used to show that any conservative system with n degrees of freedom follows Hamiltonian dynamics in a $2n$ dimensional phase space. Observing that the normalizing flow to be learned is by definition a conservative system, as it conserves the overall probability density, we can state that the HNF framework can learn the necessary dynamics, provided $\mathcal{H}_\theta$ is expressive enough.

  This last condition is in fact an open question, as we have not managed to establish optimal architecture for $\mathcal{H}_\theta$. This is especially true for image applications, where the convolutional neural networks used to map images

to scalars might suffer from the locality of the convolution operation and fail to propagate information across the images.

7.3.5 Study Conclusion

This Section introduced our proposal for the construction of continuous normalizing flows using Hamiltonian dynamics. Similar to the previous Section on unsupervised domain translation, we showed that the conservative nature of Hamiltonian dynamics could be used to construct neural flow models with desirable properties. Most notably, we have discussed the way the volume preserving nature of the HNF yielded a tractable probability density estimator, which in turns led to simplified training.

We also discussed some of the remaining issues that limit the applicability of our proposed method. Future work will be concerned with the exploration of the various improvements proposed in Section 7.3.4. We do believe that this work could lead a better overall understanding of flow generative modeling, and possibly a unified view of the problem.

Finally, we mention that the increased flexibility of the HNF could be used to model problems that would benefit from more specified architectures, for example, molecular data modeling tasks, which could benefit from architectures such as graph neural networks that might need to account for the whole data sample at every flow step.

7.4 Conclusion

With these two studies, we have proposed potential avenues to use dynamical systems theory to inform the construction of novel Deep Learning architectures. As mentioned at the start of this Chapter and in Chapter 3, approaches such as those presented above are a form of hybridization, which has given rise to various Deep Learning approaches in recent years that are now state of the art. Methods such as residual networks[82], or NeuralODEs [99] that are used extensively in this thesis are examples of such hybridization successes.

The work presented in this Chapter revolved around the use of Hamiltonian dynamical systems to construct invertible transformations. We have shown that the guarantees provided by Hamiltonian architectures yielded desirable properties. Invertibility by construction was for example used to carry out two-way domain translation in Section 7.2.2 while only training the model to learn out one-way transformations. We also mention the exploration of the volume preserving properties of Hamiltonian flows and the way they could be used to transport and compress marginal distributions.

We note that the approaches proposed in this Chapter still have significant limitations and several potential improvements should be explored to bring

them on par with the state of the art. Thus, avenues such as those proposed in Section 7.3.4 should be explored in future works.

CHAPTER 8

CONCLUSION

The application of Deep Learning methods to the improvement of numerical simulation methods has become a major research topic. As evidenced by the existence of a range of approaches to this question, some of which were introduced in the first parts of this thesis. We discussed some of the major differences between the main existing proposals, mainly from the point of view of *hybridization*, that is to say, the degree to which a method combines pre-existing physical information with data-driven modeling techniques. We tried to underline the fact that the development of efficient numerical simulation engines lies in the efficient application of *hybridization*, rather than sole reliance on either numerical or data-driven methods.

More specifically, we looked at the topic of reduced order modeling, which centers around the identification of suitable representation spaces for the dynamics of physical systems. We showed that identifying such spaces from data could help alleviate some of the cost issues associated with full order numerical simulation. However, the optimal approach to this topic is not yet established. We discussed the existence of several methods within the reduced order modeling subfield, both for dimensionality reduction and dynamical modeling, each presenting various degrees of performance and data-reliance. These considerations led to the development of two novel methods based on the hybridization of neural networks and existing reduced order modeling techniques.

First, the CD-ROM approach was introduced, a method for the development of closure models for POD-Galerkin ROMs. We derived the closure modeling architecture from considerations on the nature of the error embedded in POD-Galerkin ROMs, and dynamical systems theory. We showed that this problem was directly related to the study of partially observed systems and that capturing non-Markovian effects in the dynamics of the reduced system was critical to its accurate simulation. The proposed CD-ROM method stands as an example

of a *hybrid* model, combining information retained from the governing equations of the system, a theoretically consistent memory architecture, and a powerful data-driven closure model relying on neural networks. We also showed that this hybrid architecture could be adapted to strongly non-linear problems that would have previously been hard to reduce in the context of POD-Galerkin model reduction. Finally, we studied the way the novel continuous memory architecture of the CD-ROM selected specific frequencies to retain in memory and showed that it did so in a coherent fashion with the system to be learned.

The iLED model was then proposed for the interpretable modeling of non-linearly reduced systems. Starting from the consideration of the inefficiency of linear dimensionality reduction for dynamical systems, we developed an approach to extract interpretable dynamical laws from data. First, the well-established non-linear dimensionality reduction capabilities of neural networks were leveraged to construct very low dimensional representation spaces for various dynamical systems. We then showed that the theory of the Koopman operator introduced in the earlier stages of the thesis could be used to derive a theoretically grounded and interpretable *ansatz* for the dynamics of a physical system. We showed that systems presenting oscillatory dynamics could be reduced to very low dimensional linear systems even when governed by non-linear equations such as the Navier-Stokes equations. We also discussed the particularity of chaotic systems, such as the Kuramoto-Sivashinsky equations, which can not be modeled as purely linear systems but can still benefit from a supporting limit cycle in the latent space of an autoencoder. Once again, this proposal is placed in the context of hybrid modeling, where we used a theoretically grounded architecture to constrain a data-driven model to ensure coherence and interpretability.

8.1 Perspectives

We discuss below some of the potential applications and improvements to our proposals that should be considered in the future.

- **Control Applications:** We showed in our work that the CD-ROM and iLED methods could be used to model complex dynamical systems and extend seamlessly to parametric problems. We are now considering potential applications of our work to Model Predictive Control. This already well-established field will strongly benefit from the availability of fast and accurate models, able to extract dynamical law from a few probes measuring the state of a system. In this context, our proposals would be key enablers, opening avenues to other related research fields, such as online training, exploration, and reinforcement learning.

- **Neural Architectures:** We have centered our work around the topic of interpretability and shown that either by retaining part of the governing equations or replacing them with an easily interpretable linear term, a degree of understanding of the model could be gained. It remains that both the iLED and CD-ROM methods rely on hard-to-interpret neural closure models. We discussed in the first chapter how certain constraints could be embedded in the structure of neural networks through careful parameterization of the weight matrices. The study of the applicability of these stable parameterizations could be a first step toward understanding or at least obtaining guarantees on the behavior of these neural dynamical terms.
- **Model Extensions:** Extensions to the proposed models should be explored. As an example, the memory architecture, which is identical between both the CD-ROM and iLED approaches, could be extended to accommodate more complex memory kernels. We have discussed the advantages of a simple diagonal memory kernel as it allows for independent memory dimensions and simple memory initialization. However, more flexible parametrizations might lead to a more expressive memory architecture.

Finally, we give our perspective on the current applications of Deep Learning to numerical simulation. It is clear to us that the potential of neural networks to approximate unknown operators, such as the reduced dynamics of a system on a low dimensional manifold, can lead to major advantages and yield valuable insights into the behavior of physical systems. However, we acknowledge that the strength of numerical simulation methods lies in their generality. The representation of the real through universal laws is the foundation of physics, and this aspect of universality is clearly lacking in the existing applications of Deep Learning to simulation problems, including our proposal. This is why we believe that the development of general simulation engines that do not rely on application-specific retraining should become a major research topic in the future. Combined with ever-improving computation architectures, research in this direction could lead to orders of magnitude improvement in the design of some of our more challenging technologies.

Appendices

CHAPTER 9

APPENDIX: CD-ROM

9.1 Hyper-Parameters and training

This appendix details the various design and training choices made for the different models presented in the results section 4.6. As explained earlier in Chapter 4, the models are trained using progressively longer prediction horizons. The main advantage of this method, as opposed to directly training with the target prediction horizon, is that potentially unstable systems in the earlier learning stages will struggle to reach long-term prediction horizons without diverging, making the training extremely inefficient. Moreover, using this strategy means that a single, long DNS trajectory can be separated into numerous sub-trajectories which can be batched together and simulated in parallel, yielding a very efficient training process. The procedure is presented in algorithm 2.

Finally, the hyper-parameters values and training details for the different models trained using the above strategy are presented below. In an effort to improve readability, the various values are organized in table 9.1.

Algorithm 2 Batching and Sub-trajectories

Require: T_i the initial prediction horizon, T_f the final prediction horizon, T_{inc} the horizon increment, $\mathcal{L}_t$ the loss threshold, $f(\mathbf{z}, \theta)$ a CD-ROM model

$T \leftarrow T_i$
$\mathcal{L} \leftarrow +\infty$
while $T < T_f$ **do**
  while $\mathcal{L} > \mathcal{L}_t$ **do**
    Sample a batch of trajectories $\mathbf{a}^\star_{-\tau_{min} \rightarrow T}$
    Compute the initial memory $\mathbf{y_0}$
    $\mathbf{z_0} \leftarrow [\mathbf{a}^\star_\mathbf{0}, \mathbf{y_0}]$
    $[\mathbf{a}_t, \mathbf{y}_t] \leftarrow \mathbf{z_0} + \int_0^t f(\mathbf{z}; \theta) \mathrm{d}t$ ▷ CD-ROM simulation
    $\mathcal{L} \leftarrow \mathcal{L}(\mathbf{a}_{0 \rightarrow T}, \mathbf{a}^\star_{0 \rightarrow T})$
    Backpropagation & Gradient Step
  end while
  $T \leftarrow T + T_{inc}$
  Reorganise batches according to the new T
end while

	Cylinder	Pinball	KS
Memory Size	30	50	30
Corrector Neurons	(30,30,30,30,3)	(50,250,250,250,10)	(56,150,150,150,25)
Encoder Neurons	(3,9,15,21,27)	(10,17,24,31,40)	(26,21,16,11,5)
Activation	SiLU	SiLU	SiLU
Optimizer	Adam	Adam	Adam
Learning Rate ($\mathbf{E}$ & $\mathcal{R}$)	10^{-3}	10^{-3}	5×10^{-4}
Weight Decay ($\mathbf{E}$ & $\mathcal{R}$)	10^{-4}	10^{-2}	10^{-3}
Learning Rate ($\mathbf{\Lambda}$)	10^{-4}	2×10^{-4}	5×10^{-4}
Weight Decay ($\mathbf{\Lambda}$)	0	0	0
Time Integrator	Scipy RK-45	Scipy RK-45	Semi Implicit 3rd order [39]
Training Time	$1/2$ day	1 day	$1/2$ day

Table 9.1: Hyper parameters used in the training of the different models presented in Section 4.6. The various parts of the CD-ROM are designated as follows, $\mathcal{R}$ the neural network predicting the residual of the Galerkin model, $\mathbf{E}$ the memory encoder model, $\mathbf{\Lambda}$ the diagonal memory matrix. The final line presents the order of magnitude of the training times of each model on a single RTX 2080 gpu.

CHAPTER 10

APPENDIX: ILED

10.1 Network parameters

This Section lists the various hyperparameters and network architectures used to obtain the results presented in Section 6.4.

10.1.1 FHN

The tables below present the architecture of both the autoencoder and *iLED* dynamical models used to obtain the results on the FHN case.

Layer	Encoder
(1)	ConstantPad1d(padding=(13, 14), value=0.0)
(2)	Conv1d(2, 8, kernel_size=(5,), stride=(1,), padding=same)
(3)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(4)	SiLU()
(5)	Conv1d(8, 16, kernel_size=(5,), stride=(1,), padding=same)
(6)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(7)	SiLU()
(8)	Conv1d(16, 32, kernel_size=(5,), stride=(1,), padding=same)
(9)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(10)	SiLU()
(11)	Conv1d(32, 4, kernel_size=(5,), stride=(1,), padding=same)
(12)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(13)	SiLU()
(14)	Flatten(start_dim=-2, end_dim=-1)
(15)	Linear(in_features=32, out_features=2, bias=True)
(16)	LatentSpaceCenteringLayer()
Layer	**Decoder**
(1)	Linear(in_features=2, out_features=32, bias=True)
(2)	SiLU()
(3)	Unflatten(dim=-1, unflattened_size=(4, 8))
(4)	Upsample(scale_factor=2.0, mode=linear)
(5)	ConvTranspose1d(4, 32, kernel_size=(5,), stride=(1,), padding=(2,))
(6)	SiLU()
(7)	Upsample(scale_factor=2.0, mode=linear)
(8)	ConvTranspose1d(32, 16, kernel_size=(5,), stride=(1,), padding=(2,))
(9)	SiLU()
(10)	Upsample(scale_factor=2.0, mode=linear)
(11)	ConvTranspose1d(16, 8, kernel_size=(5,), stride=(1,), padding=(2,))
(12)	SiLU()
(13)	Upsample(scale_factor=2.0, mode=linear)
(14)	ConvTranspose1d(8, 2, kernel_size=(5,), stride=(1,), padding=(2,))
(15)	1 + 0.5 Tanh()
(16)	Unpad()

Table 10.1: One-dimensional convolutional autoencoder used to obtain the results on the case of the FHN model presented in Section 6.4.1

	iLED Parameters
$\mathbf{A}_\theta$	Linear(2,2,bias=False)
$\mathbf{\Psi}_1$ neurons	18 - 32 - 32 - 32 - 2
$\mathbf{\Psi}_1$ activation	SiLU()
d_h	16
$\mathbf{\Psi_2}$	AugmentedIdentityEncoder (see Eq. (6.29))
$\mathbf{\Psi_2}$ neurons	2 - 5 - 8 - 11 - 14
$\mathbf{\Psi_2}$ activation	SiLU()
$\mathbf{\Lambda}_\theta$	diag$(\boldsymbol{w})$, $\boldsymbol{w} \in \mathbb{R}^{d_h}_-$

Table 10.2: Hyperparameters of the *iLED* dynamics used to obtain the results on the FHN case presented in Section 6.4.1.

10.1.2 KS

Similar to the previous paragraph, the architecture of the networks used for the KS case are presented below.

Layer	Encoder
(1)	Conv1d(1, 16, kernel_size=(5,), stride=(1,), padding=same)
(2)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(2), (0,))
(3)	SiLU()
(4)	Conv1d(16, 32, kernel_size=(5,), stride=(1,), padding=same)
(5)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(6)	SiLU()
(7)	Conv1d(32, 64, kernel_size=(5,), stride=(1,), padding=same)
(8)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(9)	SiLU()
(10)	Conv1d(64, 8, kernel_size=(5,), stride=(1,), padding=same)
(11)	AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
(12)	SiLU()
(13)	Flatten(start_dim=-2, end_dim=-1)
(14)	Linear(in_features=64, out_features=8, bias=True)
(15)	LatentSpaceCentering()
Layer	**Decoder**
(1)	Linear(in_features=8, out_features=64, bias=True)
(2)	Unflatten(dim=-1, unflattened_size=(8, 8))
(3)	Upsample(scale_factor=2.0, mode=linear)
(4)	ConvTranspose1d(8, 64, kernel_size=(5,), stride=(1,), padding=(2,))
(5)	SiLU()
(6)	Upsample(scale_factor=2.0, mode=linear)
(7)	ConvTranspose1d(64, 32, kernel_size=(5,), stride=(1,), padding=(2,))
(8)	SiLU()
(9)	Upsample(scale_factor=2.0, mode=linear)
(10)	ConvTranspose1d(32, 16, kernel_size=(5,), stride=(1,), padding=(2,))
(11)	SiLU()
(12)	Upsample(scale_factor=2.0, mode=linear)
(13)	ConvTranspose1d(16, 1, kernel_size=(5,), stride=(1,), padding=(2,))

Table 10.3: One-dimensional convolutional autoencoder used to obtain the results on the case of the KS equation (sec 6.4.2)

	iLED Parameters
$\mathbf{A}_\theta$	$\mathbf{W} - \mathbf{W}^T - \text{diag}(\boldsymbol{w})$, $\mathbf{W} \in \mathbb{R}^{d_z \times d_z}, \boldsymbol{w} \in \mathbb{R}_+^{d_z}$
$\boldsymbol{\Psi}_1$ neurons	40 - 64 - 64 - 64 - 8
$\boldsymbol{\Psi}_1$ activation	SiLU()
d_h	32
$\boldsymbol{\Psi_2}$	AugmentedIdentityEncoder (see Eq. (6.29))
$\boldsymbol{\Psi_2}$ neurons	8 - 12 - 16 - 20 - 24
$\boldsymbol{\Psi_2}$ activation	SiLU()
$\boldsymbol{\Lambda}_\theta$	$\text{diag}(\boldsymbol{w})$, $\boldsymbol{w} \in \mathbb{R}_-^{d_h}$

Table 10.4: Hyperparameters of the *iLED* dynamics used to obtain the results on the KS case presented in Section 6.4.2.

10.1.3 Flow around a cylinder

The autoencoders used in the Cylinder flow case have a complex architecture, to simplify the notation, we define two blocks that combine similar operations:

Table 10.5: Sub blocks defined to help describe the CNN autoencoders

Layer	DownBlock(in_size,out_size)
(1)	Conv2d(in_size, out_size, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), padding_mode=replicate)
(2)	SiLU()
Layer	**UpBlock(in_size,out_size)**
(1)	Upsample(scale_factor=2.0, mode=bilinear)
(2)	Conv2d(in_size, out_size, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), padding_mode=replicate)
(3)	SiLU()
(4)	BatchNorm2d()

Layer	Encoder #1	Encoder #2
(1)	DownBlock(2,4)	DownBlock(2,4)
(2)	DownBlock(4,16)	DownBlock(4,8)
(3)	DownBlock(16,4)	DownBlock(8,16)
(4)	DownBlock(4,2)	DownBlock(16,2)
(5)	Flatten(start=-3,end=-1)	Flatten(start=-3,end=-1)
(6)	Linear(512,20)	Linear(200,20)
(7)	$d_{z_1} = 20$	$d_{z_2} = 20$
	Mixer Encoder	
(8)	Concatenate(z_1, z_2)	
(9)	Linear(40,30)	
(10)	SiLU()	
(11)	Linear(30,d_z)	
Layer	Mixer Decoder	
(1)	Linear(d_z,30)	
(2)	SiLU()	
(3)	Linear(30,40)	
(4)	$z_1, z_2 = z$	
	Decoder #1	Decoder #2
(5)	Linear(20,512)	Linear(20,200)
(6)	Unflatten(-1,(2,32,8))	Unflatten(-1,(2,10,10)
(7)	UpBlock(2,4)	UpBlock(2,16)
(8)	UpBlock(4,16)	UpBlock(16,8)
(9)	UpBlock(16,4)	UpBlock(8,4)
(10)	Upsample(2.0,bilinear)	Upsample(2.0,bilinear)
(11)	Conv2d(4, 1, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), padding_mode=replicate)	Conv2d(4, 1, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), padding_mode=replicate)
(12)	Flatten(start=-3,end=-1)	Flatten(start=-3,end=-1)
(13)	StreamFnToVelocity()	StreamFnToVelocity()

Table 10.6: Hyperparameters of the 2-dimensional convolutional autoencoder used to obtain the results on the Cylinder case presented in Section 6.4.3.

The value of d_z changes depending on the Reynolds number considered. It is equal to $d_z = 3$ in the $Re = 100$ case, and $d_z = 16$ in the $Re = 750$ case. Note that we don't use a *LatentSpaceCentering* layer contrary to the other cases. This is due to the fact that, because of the memory costs of the models, the batch size has to be relatively low, which has a negative impact on batch normalization approaches. To ensure that the latent space remained centered, which is critical to the accuracy and interpretability of the linear term in the dynamics, we added

a term to the loss:

$$\mathcal{L}_{centering} = \| \frac{1}{N_T} \sum_{i=1}^{N_T} \mathcal{E}(\mathbf{\Phi}_{t_i}) \|_2^2. \tag{10.1}$$

This loss effectively penalizes the average of the latent codes, ensuring that they are centered around the origin.

Also note that the decoder doesn't directly predict the velocity field, but the stream function ψ which is a scalar field, that is used to compute the velocity components as follows:

$$u_x = \frac{\partial \psi}{\partial y}, \; u_y = -\frac{\partial \psi}{\partial x}. \tag{10.2}$$

This specific choice is inspired from previous works ([141, 189]). It allows for the guaranteed prediction of a divergence free field, which is a constraint of the incompressible Navier-Stokes equations.

Finally, the parameters of the *iLED* dynamical model are provided below:

	iLED Parameters
$\mathbf{A}_\theta$	$\mathbf{W} - \mathbf{W}^T - \text{diag}(\boldsymbol{w})$, $\mathbf{W} \in \mathbb{R}^{d_z \times d_z}, \boldsymbol{w} \in \mathbb{R}_+^{d_z}$
$\mathbf{\Psi}_1$ neurons	$d_z + d_h$ - 128 - 128 - 128 - d_z
$\mathbf{\Psi}_1$ activation	SiLU()
d_h	32
$\mathbf{\Psi_2}$	AugmentedIdentityEncoder (see Eq. (6.29))
$\mathbf{\Psi_2}$ neurons ($Re = 750$)	16 - 17 - 17 - 17 - 15
$\mathbf{\Psi_2}$ neurons ($Re = 100$)	3 - 10 - 16 - 22 - 28
$\mathbf{\Psi_2}$ activation	SiLU()
$\mathbf{\Lambda}_\theta$	$\text{diag}(\boldsymbol{w})$, $\boldsymbol{w} \in \mathbb{R}_-^{d_h}$

Table 10.7: Hyperparameters of the *iLED* dynamics used to obtain the results on the Cylinder cases presented in Section 6.4.3.

CHAPTER 11

SYNTHÈSE EN FRANÇAIS

Les systèmes dynamiques sont généralement modélisés à l'aide d'équations aux dérivées partielles (EDP). Ces modèles sont étroitement liés à la façon dont les scientifiques observent le monde et, en tant que tels, ils sont limités par notre compréhension des systèmes étudiés. En effet, des modèles tels que les équations de Navier-Stokes ne modélisent que les interactions locales dans un écoulement, et négligent les phénomènes sous-jacents qui controlent le système dans son ensemble. Ce caractère local des modèles basés sur les EDP couplé à la complexité des phénomènes et géométries étudiés dans les applications industrielles implique l'utilisation de discrètisations très fines pour la simulation des phénomènes physiques. Ce qui conduit généralement à des coûts de calcul excessifs associés à la résolution numérique des EDP.

Dans cette thèse, nous discutons de la manière dont les données peuvent être exploitées pour dériver de meilleurs espaces de représentation pour les systèmes physiques ainsi que des modèles dynamiques simplifiés, appelés modèles réduits. Nous présentons d'abord quelques unes des approches de réduction de modèle existantes. Nous proposons ensuite d'exploiter les capacités d'approximation des réseaux de neurones pour construire de nouvelles méthodes de réduction de modèles. Les approches introduites dans cette thèse reposent sur le concept d'hybridation entre la modélisation physique et les méthodes d'apprentissage machine. Nous nous appuyons sur les propriétés des systèmes dynamiques étudiés pour construire des modèles interprétables, précis et en accord avec la théorie afin de résoudre les problèmes de coûts de calcul associés à la modélisation physique standard, tout en limitant la dépendance des modèles aux données. Les travaux présentés dans cette thèse peuvent être séparés en deux propositions distinctes.

CD-ROM : La méthode *Complemented Deep - Reduced Order Model* propose une approche de fermeture basée sur les réseaux de neurones pour les mod-

èles à ordre réduit de type POD-Galerkin. L'approche est basée sur la théorie des systèmes partiellement observés, elle utilise des réseaux de neurones pour approximer les erreurs inhérentes aux modèle réduits POD-Galerkin, tout en conservant une partie des équations qui gouvernent le système. Contrairement à la plupart des travaux précédents sur la réduction de modèle à l'aide des réseaux neuronaux, l'approche CD-ROM est basée sur une formulation de mémoire interprétable et continue en temps, dérivée d'hypothèses simples sur le comportement des systèmes dynamiques. Cette formulation continue en temps permet de simuler les modèles construits avec la méthode CD-ROM à l'aide d'intégrateurs temporels standards, là où les réseaux de neurones récurrents reposent généralement sur une progression discrète en temps.

iLED : La méthode *intepretable Learning of Effective Dynamics for multiscale systems* est une approche de modélisation dynamique entièrement basée sur les données. La méthode propose d'utiliser les réseaux de neurones pour construire à la fois une représentation du système étudié en dimension réduite et un modèle dynamique pour la simulation du système dans l'espace réduit obtenu. Nous montrons que iLED offre une précision comparable aux approches basées sur les réseaux de neurones récurrents tout en retenant un degré d'interprétabilité élevé. L'architecture iLED est directement basée sur la théorie de l'opérateur de Koopman. Nous montrons dans nos expériences que iLED peut être utilisé pour dériver des équations quasi linéaires de basse dimension pour des EDP non linéaires généralement résolues dans des espaces de très haute dimension, fournissant des informations précieuses sur la dynamique étudiée et réduisant considérablement les coûts de calcul associés à la simulation du modèle.

Chaque méthode est illustrée à l'aide de cas standard de la littérature, tels que les écoulements fluides bidimensionnels et les systèmes chaotiques tels que les équations de Kuramoto-Sivashinsky.

BIBLIOGRAPHY

[1] B. O. Koopman. "Hamiltonian Systems and Transformations in Hilbert Space". In: *Proceedings of the National Academy of Sciences of the United States of America* 17.5 (1931), pp. 315–318. issn: 0027-8424. url: https://www.jstor.org/stable/86114 (visited on 12/30/2019).

[2] M. Rosenblatt. "Remarks on a Multivariate Transformation". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 470–472. doi: 10.1214/aoms/1177729394. url: https://doi.org/10.1214/aoms/1177729394.

[3] P. D. Lax and R. D. Richtmyer. "Survey of the stability of linear finite difference equations". In: *Communications on Pure and Applied Mathematics* 9.2 (1956), pp. 267–293. doi: https://doi.org/10.1002/cpa.3160090206. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.3160090206. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160090206.

[4] L. Pontriagin et al. *The Mathematical Theory of Optimal Processes*. Interscience publishers. Interscience Publishers, 1962. isbn: 9780470693810.

[5] A. Tikhonov. "Solution of Incorrectly Formulated Problems and the Regularization Method." In: *Soviet Mathematics Doklady* (1963), pp. 1035–1038.

[6] H. Mori. "Transport, Collective Motion, and Brownian Motion". In: *Progress of Theoretical Physics* 33.3 (Mar. 1965), pp. 423–455. doi: 10.1143/PTP.33.423.

[7] S. Amari. "A Theory of Adaptive Pattern Classifiers". In: *IEEE Transactions on Electronic Computers* EC-16.3 (1967), pp. 299–307. doi: 10.1109/PGEC.1967.264666.

[8] J. L. Lumley. “The structure of inhomogeneous turbulent flows”. In: *Atmospheric turbulence and radio wave propagation* (1967).

[9] J. L. Lumley. *Stochastic tools in turbulence. volume 12. applied mathematics and mechanics*. Tech. rep. Pennsylvania state university, dept of aerospace engineering, 1970.

[10] R. Zwanzig. “Nonlinear generalized Langevin equations”. In: *Journal of Statistical Physics* (1973). url: https://doi.org/10.1007/BF01008729.

[11] J. Dormand and P. Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of Computational and Applied Mathematics* 6.1 (1980), pp. 19–26. issn: 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3.

[12] F. Takens. “Detecting strange attractors in turbulence”. In: *Dynamical Systems and Turbulence, Warwick 1980*. Ed. by D. Rand and L.-S. Young. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 366–381. isbn: 978-3-540-38945-3.

[13] P. Grassberger and I. Procaccia. “Measuring the strangeness of strange attractors”. In: *Physica D: Nonlinear Phenomena* 9.1 (1983), pp. 189–208. issn: 0167-2789. doi: https://doi.org/10.1016/0167-2789(83)90298-1.

[14] A. Wolf et al. “Determining Lyapunov exponents from a time series”. In: *Physica D: Nonlinear Phenomena* 16.3 (1985), pp. 285–317. issn: 0167-2789. doi: https://doi.org/10.1016/0167-2789(85)90011-9.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[16] Y. LeCun et al. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551. doi: 10.1162/neco.1989.1.4.541.

[17] A. E. Deane et al. “Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders”. In: *Physics of Fluids* 3 (1991), pp. 2337–2354.

[18] K. Gopalsamy. *Stability and Oscillations in Delay Differential Equations of Population Dynamics*. Kluwer Academic Publishers, 1992. isbn: 9780470693810.

[19] A. Mead. “Review of the Development of Multidimensional Scaling Methods”. In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 41.1 (1992), pp. 27–39. issn: 00390526, 14679884.

[20] M. Leshno et al. “Multilayer feedforward networks with a non-polynomial activation function can approximate any function”. In: *Neural Networks* 6.6 (1993), pp. 861–867. issn: 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80131-5.

[21] M. Schoenauer and E. Ronald. “Neuro-Genetic Truck Backer-Upper Controller”. In: *Proc. First IEEE International Conference on Evolutionary Computation*. Orlando, United States: IEEE, June 1994. url: https://inria.hal.science/hal-02985436.

[22] T. Chen and H. Chen. “Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems”. In: *IEEE Transactions on Neural Networks* 6.4 (1995), pp. 911–917.

[23] S. Hochreiter and J. Schmidhuber. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. doi: 10.1162/neco.1997.9.8.1735.

[24] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[25] E. E. Salpeter and S. R. Salpeter. “Mathematical Model for the Epidemiology of Tuberculosis, with Estimates of the Reproductive Number and Infection-Delay Function”. In: *American Journal of Epidemiology* 147.4 (Feb. 1998), pp. 398–406. issn: 0002-9262. doi: 10.1093/oxfordjournals.aje.a009463.

[26] B. Schölkopf, A. Smola, and K.-R. Müller. “Nonlinear Component Analysis as a Kernel Eigenvalue Problem”. In: *Neural Computation* 10.5 (July 1998), pp. 1299–1319. issn: 0899-7667. doi: 10.1162/089976698300017467. eprint: https://direct.mit.edu/neco/article-pdf/10/5/1299/813905/089976698300017467.pdf. url: https://doi.org/10.1162/089976698300017467.

[27] D. Opitz and R. Maclin. “Popular Ensemble Methods: An Empirical Study”. In: *Journal of Artificial Intelligence Research* 11 (1999), pp. 169–198.

[28] S. Chen and R. Gopinath. “Gaussianization”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. url: https://proceedings.neurips.cc/paper/2000/file/3c947bc2f7ff007b86a9428b74654de5-Paper.pdf.

[29] R. Zwanzig. *Nonequilibrium statistical mechanics*. Oxford university press, 2001.

[30] M. Milano and P. Koumoutsakos. "Neural Network Modeling for Near Wall Turbulent Flow". In: *Journal of Computational Physics* 182.1 (2002), pp. 1–26. issn: 0021-9991. doi: https://doi.org/10.1006/jcph.2002.7146. url: https://www.sciencedirect.com/science/article/pii/S0021999102971469.

[31] G. Berkooz, P. Holmes, and J. Lumley. "The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows". In: *Annual Review of Fluid Mechanics* 25 (Nov. 2003), pp. 539–575. doi: 10.1146/annurev.fl.25.010193.002543.

[32] B. Noack et al. "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake". In: *Journal of Fluid Mechanics* 497 (Dec. 2003), pp. 335–363. doi: 10.1017/S0022112003006694.

[33] C. Pruett et al. "The temporally filtered Navier–Stokes equations: properties of the residual stress". In: *Physics of Fluids* 15.8 (2003), pp. 2127–2140.

[34] R. Coifman et al. "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps". In: *PNAS* (2005). url: https://doi.org/10.1073/pnas.0500334102.

[35] C. W. Rowley. "Model Reduction for fluids, Using Balanced Proper Orthogonal Decomposition". In: *Int. J. Bifurc. Chaos* 15 (2005), pp. 997–1013.

[36] E. Åkervik et al. "Steady solutions of the Navier-Stokes equations by selective frequency damping". In: *Physics of fluids* 18.6 (2006), p. 068102.

[37] J. Burkardt, M. Gunzburger, and H.-C. Lee. "POD and CVT-based reduced-order modeling of Navier–Stokes flows". In: *Computer Methods in Applied Mechanics and Engineering* 196.1 (2006), pp. 337–355. issn: 0045-7825. doi: https://doi.org/10.1016/j.cma.2006.04.004. url: https://www.sciencedirect.com/science/article/pii/S0045782506001575.

[38] K. I. V. et al. "Elements of the Lattice Boltzmann Method I: Linear Advection Equation". In: *Communications in Computational Physics* 1.4 (2006), pp. 616–655. issn: 1991-7120. doi: https://doi.org/. url: http://global-sci.org/intro/article_detail/cicp/7972.html.

[39] S. K. Kar. "A Semi-Implicit Runge–Kutta Time-Difference Scheme for the Two-Dimensional Shallow-Water Equations". In: *Monthly Weather Review* 134.10 (2006), pp. 2916–2926. doi: 10.1175/MWR3214.1.

[40] A. Chorin and P. Stinis. "Problem reduction, renormalization, and memory". In: *Communications in Applied Mathematics and Computational Science* 1.1 (2007), pp. 1–27.

[41] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. url: http://jmlr.org/papers/v9/vandermaaten08a.html.

[42] P. Schmid and J. Sesterhenn. "Dynamic Mode Decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656 (Nov. 2008). doi: 10.1017/S0022112010001217.

[43] S. Chaturantabut and D. C. Sorensen. "Discrete Empirical Interpolation for nonlinear model reduction". In: (2009), pp. 4316–4321. doi: 10.1109/CDC.2009.5400045.

[44] E. Darve, J. Solomon, and A. Kia. "Computing generalized Langevin equations and generalized Fokker–Planck equations". In: *Proceedings of the National Academy of Sciences* 106.27 (2009), pp. 10884–10889.

[45] M. Lukosevicius and H. Jaeger. "Reservoir computing approaches to recurrent neural network training". In: *Comput. Sci. Rev.* 3 (2009), pp. 127–149.

[46] D. Amsallem and C. Farhat. "An Online Method for Interpolating Linear Parametric Reduced-Order Models". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2169–2198. doi: 10.1137/100813051. eprint: https://doi.org/10.1137/100813051. url: https://doi.org/10.1137/100813051.

[47] M. Budišić, R. Mohr, and I. Mezić. "Applied koopmanism". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.4 (2012), p. 047510.

[48] P. Holmes et al. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. url: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[50] A. Logg, K. Mardal, and G. Wells. *Automated Solution of Differential Equations by the Finite Element Method*. 2012. doi: https://doi.org/10.1007/978-3-642-23099-8.

[51] R. Pascanu, T. Mikolov, and Y. Bengio. "On the difficulty of training Recurrent Neural Networks". In: *30th International Conference on Machine Learning, ICML 2013* (Nov. 2012).

[52] T. Tieleman and G. Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[53] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. 2013. doi: 10.48550/ARXIV.1312.6114. url: https://arxiv.org/abs/1312.6114.

[54] M. Lichman. *UCI machine learning repository*. 2013. url: http://archive.ics.uci.edu/ml..

[55] I. Mezić. "Analysis of Fluid Flows via Spectral Properties of the Koopman Operator". In: *Annual Review of Fluid Mechanics* 45.1 (2013), pp. 357–378. doi: 10.1146/annurev-fluid-011212-140652.

[56] K. Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (June 2014). doi: 10.3115/v1/D14-1179.

[57] J. Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: (Dec. 2014).

[58] L. Dinh, D. Krueger, and Y. Bengio. *NICE: Non-linear Independent Components Estimation*. 2014. doi: 10.48550/ARXIV.1410.8516. url: https://arxiv.org/abs/1410.8516.

[59] D. P. G. Foures et al. "A data-assimilation method for Reynolds-averaged Navier–Stokes-driven mean flow reconstruction". In: *Journal of Fluid Mechanics* 759 (2014), pp. 404–431. doi: 10.1017/jfm.2014.566.

[60] I. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. url: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[61] I. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. url: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[62] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. "Sparsity-promoting dynamic mode decomposition". In: *Physics of Fluids* 26.2 (2014), p. 024103. doi: 10.1063/1.4863670.

[63] E. Kaiser et al. “Cluster-based reduced-order modelling of a mixing layer”. In: *Journal of Fluid Mechanics* 754 (Aug. 2014), pp. 365–414. issn: 1469-7645. doi: 10.1017/jfm.2014.355.

[64] T. Lassila et al. “Model order reduction in fluid dynamics: challenges and perspectives”. In: *Reduced Order Methods for modeling and computational reduction* (2014), pp. 235–273.

[65] N. Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. url: http://jmlr.org/papers/v15/srivastava14a.html.

[66] J. H. Tu et al. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. doi: 10.3934/jcd.2014.1.391.

[67] M. Alnaes et al. *The FEniCS Project*. 2015. doi: https://doi.org/10.11588/ans.2015.100.20553.

[68] P. Benner, S. Gugercin, and K. Willcox. “A survey of projection-based model reduction methods for parametric dynamical systems”. In: *SIAM review* 57.4 (2015), pp. 483–531.

[69] D. Kondrashov, M. D. Chekroun, and M. Ghil. “Data-driven non-Markovian closure models”. In: *Physica D: Nonlinear Phenomena* 297 (2015), pp. 33–55.

[70] Z. Liu et al. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[71] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. url: https://www.tensorflow.org/.

[72] I. G. K. Matthew O. Williams Clarence W. Rowley. “A kernel-based method for data-driven koopman spectral analysis”. In: *Journal of Computational Dynamics* 2.2 (2015), pp. 247–265.

[73] B. Peherstorfer and K. Willcox. “Dynamic data-driven reduced-order models”. In: *Computer Methods in Applied Mechanics and Engineering* 291 (2015), pp. 21–41. issn: 0045-7825. doi: https://doi.org/10.1016/j.cma.2015.03.018. url: https://www.sciencedirect.com/science/article/pii/S0045782515001280.

[74] D. Rezende and S. Mohamed. “Variational Inference with Normalizing Flows”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1530–1538. url: https://proceedings.mlr.press/v37/rezende15.html.

[75] J. Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2256–2265. url: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[76] N. Srivastava, E. Mansimov, and R. Salakhudinov. “Unsupervised learning of video representations using lstms”. In: *International conference on machine learning*. PMLR. 2015, pp. 843–852.

[77] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. “A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition”. In: *Journal of Nonlinear Science* 25.6 (Dec. 2015), pp. 1307–1346. issn: 1432-1467. doi: 10.1007/s00332-015-9258-5.

[78] A. Botev, G. Lever, and D. Barber. *Nesterov’s Accelerated Gradient and Momentum as approximations to Regularised Update Descent*. 2016. arXiv: 1607.01981 [stat.ML].

[79] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. issn: 0027-8424. doi: 10.1073/pnas.1517384113. url: https://www.pnas.org/content/113/15/3932.

[80] L. A. Gatys, A. S. Ecker, and M. Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423. doi: 10.1109/CVPR.2016.265.

[81] G. Haller and S. Ponsioen. *Nonlinear normal modes and spectral submanifolds: Existence, uniqueness and use in model reduction*. 2016. arXiv: 1602.00560 [math.DS].

[82] K. He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[83] H. P. Langtangen and K.-A. Mardal. *Introduction to Numerical Methods for Variational Problems*. 2016. url: http://hplgit.github.io/fem-book/doc/pub/book/pdf/fem-book-4screen.pdf.

[84] K. Carlberg, M. Barone, and H. Antil. “Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction”. In: *Journal of Computational Physics* 330 (2017), pp. 693–734. issn: 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.10.033.

[85] L. Dinh, J. Sohl-Dickstein, and S. Bengio. "Density estimation using Real NVP". In: *International Conference on Learning Representations*. 2017. url: https://openreview.net/forum?id=HkpbnH9lx.

[86] E. Haber and L. Ruthotto. "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (Dec. 2017), p. 014004. issn: 1361-6420. doi: 10.1088/1361-6420/aa9a90.

[87] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[88] Q. Li et al. "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.10 (Oct. 2017), p. 103111. issn: 1089-7682. doi: 10.1063/1.4993854.

[89] B. R. Noack and M. Morzyński. *The fluidic pinball — a toolkit for multiple-input multiple-output flow control*. Tech. rep. Chair of Virtual Engineering, Poznan University of Technology, Poland, 2017.

[90] S. Otto and C. Rowley. "Linearly-Recurrent Autoencoder Networks for Learning Dynamics". In: *SIAM Journal on Applied Dynamical Systems* 18 (Dec. 2017). doi: 10.1137/18M1177846.

[91] G. Papamakarios, T. Pavlakou, and I. Murray. "Masked Autoregressive Flow for Density Estimation". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[92] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations". In: *arXiv preprint arXiv:1711.10561* (2017).

[93] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations". In: *arXiv preprint arXiv:1711.10566* (2017).

[94] C. W. Rowley and S. T. Dawson. "Model Reduction for Flow Analysis and Control". In: *Annual Review of Fluid Mechanics* 49.1 (2017), pp. 387–417. doi: 10.1146/annurev-fluid-010816-060042. url: https://doi.org/10.1146/annurev-fluid-010816-060042.

[95] J.-Y. Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251. doi: 10.1109/ICCV.2017.244.

[96] A. G. Baydin et al. "Automatic Differentiation in Machine Learning: a Survey". In: *Journal of Machine Learning Research* 18.153 (2018), pp. 1–43. url: http://jmlr.org/papers/v18/17-468.html.

[97] J. Berg and K. Nyström. "A unified deep ANN approach to PDEs in complex geometries". In: *Neurocomputing* 317 (2018), pp. 28–41.

[98] J. Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. url: http://github.com/google/jax.

[99] R. T. Q. Chen et al. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[100] G. Y. Cornejo Maceda et al. "Taming the fluidic pinball with artificial intelligence control". In: *European Fluid Mechanics Conference*. Vienne, Austria, Sept. 2018.

[101] C. Donahue, J. McAuley, and M. Puckette. *Adversarial Audio Synthesis*. 2018. doi: 10.48550/ARXIV.1802.04208. url: https://arxiv.org/abs/1802.04208.

[102] N. B. Erichson et al. *Diffusion Maps meet Nyström*. 2018. arXiv: 1802.08762 [stat.ML].

[103] D. P. Kingma and P. Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. doi: 10.48550/ARXIV.1807.03039. url: https://arxiv.org/abs/1807.03039.

[104] H. Lin and S. Jegelka. *ResNet with one-neuron hidden layers is a Universal Approximator*. 2018. arXiv: 1806.10909 [cs.LG].

[105] J.-C. Loiseau and S. L. Brunton. "Constrained sparse Galerkin regression". In: *Journal of Fluid Mechanics* 838 (2018), pp. 42–67.

[106] B. Lusch, J. N. Kutz, and S. L. Brunton. "Deep learning for universal linear embeddings of nonlinear dynamics". In: *Nature Communications* 9.1 (2018). doi: 10.1038/s41467-018-07210-0. url: https://doi.org/10.1038%2Fs41467-018-07210-0.

[107] M. Mannattil. *NoLiTSA*. https://github.com/charlespwd/project-title. 2018.

[108] T. Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations*. 2018. url: https://openreview.net/forum?id=B1QRgziT-.

[109] M. Raissi. "Deep hidden physics models: Deep learning of nonlinear partial differential equations". In: *JMLR* 19.1 (2018), pp. 932–955.

[110] J. Sirignano and K. Spiliopoulos. "DGM: A deep learning algorithm for solving PDEs". In: *Journal of computational physics* 375 (2018), pp. 1339–1364.

[111] A. F. Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2019. arXiv: 1803.08375 [cs.NE].

[112] L. Ardizzone et al. *Analyzing Inverse Problems with Invertible Neural Networks*. 2019. arXiv: 1808.04730 [cs.LG].

[113] J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. "Invertible Residual Networks". In: *ICML*. 2019.

[114] K. P. Champion, S. L. Brunton, and J. N. Kutz. "Discovery of nonlinear multiscale systems: Sampling strategies and embeddings". In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 312–333.

[115] Y. Choi and K. Carlberg. "Space–time least-squares Petrov–Galerkin projection for nonlinear model reduction". In: *SIAM Journal on Scientific Computing* 41.1 (2019), A26–A58.

[116] N. Deng et al. "Low-order model for successive bifurcations of the fluidic pinball". In: *Journal of Fluid Mechanics* 884 (Dec. 2019). issn: 1469-7645. doi: 10.1017/jfm.2019.959.

[117] W. Grathwohl et al. "Scalable Reversible Generative Models with Free-form Continuous Dynamics". In: *International Conference on Learning Representations*. 2019. url: https://openreview.net/forum?id=rJxgknCcK7.

[118] S. Greydanus, M. Dzamba, and J. Yosinski. "Hamiltonian Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[119] J. Ho et al. "Flow++: Improving flow-based generative models with variational dequantization and architecture design". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2722–2730.

[120] T. Karras, S. Laine, and T. Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[121] M. M. N. B. Erichson and M. Mahoney. "Physics-informed Autoencoders for Lyapunov-stable Fluid Flow Prediction". In: *Machine Learning and the Physical Sciences Workshop, Conference on Neural Information Processing Systems*. 2019. url: https://arxiv.org/abs/1905.10866.

[122] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. url: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[123] S. Pawar et al. “A deep learning enabler for nonintrusive reduced order modeling of fluid flows”. In: *Physics of Fluids* 31.8 (2019), p. 085101. doi: 10.1063/1.5113494.

[124] 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”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[125] A. Razavi et al. *Preventing Posterior Collapse with delta-VAEs*. 2019. doi: 10.48550/ARXIV.1901.03416. url: https://arxiv.org/abs/1901.03416.

[126] S. Zhao, J. Song, and S. Ermon. “InfoVAE: Balancing Learning and Inference in Variational Autoencoders”. In: *AAAI Conference on Artificial Intelligence*. 2019.

[127] L. Biferale et al. *TURB-Rot. A large database of 3d and 2d snapshots from turbulent rotating flows*. 2020. arXiv: 2006.07469 [physics.flu-dyn].

[128] N. Doan, W. Polifke, and L. Magri. “Physics-informed echo state networks”. In: *Journal of Computational Science* 47 (Nov. 2020), p. 101237. issn: 1877-7503. doi: 10.1016/j.jocs.2020.101237.

[129] H. Eivazi et al. “Deep neural networks for nonlinear model order reduction of unsteady flows”. In: *Physics of Fluids* 32.10 (2020), p. 105104. doi: 10.1063/5.0020526.

[130] F. Farnia and A. Ozdaglar. “Do GANs always have Nash equilibria?” In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 3029–3039. url: https://proceedings.mlr.press/v119/farnia20a.html.

[131] H. Gao, J.-X. Wang, and M. Zahr. “Non-intrusive model reduction of large-scale, nonlinear dynamical systems using deep learning”. In: *Physica D: Nonlinear Phenomena* 412 (June 2020), p. 132614. doi: 10.1016/j.physd.2020.132614.

[132] L. Girin et al. “Dynamical variational autoencoders: A comprehensive review”. In: *arXiv preprint arXiv:2008.12595* (2020).

[133] C.-W. Huang, L. Dinh, and A. C. Courville. "Augmented Normalizing Flows: Bridging the Gap Between Generative Flows and Latent Variable Models". In: *ArXiv* abs/2002.07101 (2020).

[134] C.-H. Lee et al. "Maskgan: Towards diverse and interactive facial image manipulation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5549–5558.

[135] K. Lee and K. T. Carlberg. "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders". In: *Journal of Computational Physics* 404 (2020), p. 108973.

[136] X. Li et al. "Scalable Gradients for Stochastic Differential Equations". In: *arXiv preprint arXiv:2001.01328* (2020).

[137] Z. Li et al. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[138] Z. Li et al. "Neural operator: Graph kernel network for partial differential equations". In: *arXiv preprint arXiv:2003.03485* (2020).

[139] J.-C. Loiseau. "Data-driven modeling of the chaotic thermal convection in an annular thermosyphon". In: *Theoretical and Computational Fluid Dynamics* 34.4 (2020), pp. 339–365. doi: 10.1007/s00162-020-00536-w. url: https://doi.org/10.1007%2Fs00162-020-00536-w.

[140] R. Maulik et al. "Time-series learning of latent-space dynamics for reduced-order model closure". In: *Physica D: Nonlinear Phenomena* 405 (Apr. 2020), p. 132368. doi: 10.1016/j.physd.2020.132368.

[141] A. T. Mohan et al. *Embedding Hard Physical Constraints in Neural Network Coarse-Graining of 3D Turbulence*. 2020. arXiv: 2002.00021 [physics.comp-ph].

[142] S. Pan and K. Duraisamy. "Physics-Informed Probabilistic Learning of Linear Embeddings of Nonlinear Dynamics with Guaranteed Stability". In: *SIAM Journal on Applied Dynamical Systems* 19.1 (Jan. 2020), pp. 480–509. issn: 1536-0040. doi: 10.1137/19m1267246.

[143] S. Pawar et al. "Data-driven recovery of hidden physics in reduced order modeling of fluid flows". In: *Physics of Fluids* 32.3 (2020), p. 036602. doi: 10.1063/5.0002051.

[144] L. Plasman, J. Deteix, and D. Yakoubi. “A projection scheme for Navier-Stokes with variable viscosity and natural boundary condition”. In: *International Journal for Numerical Methods in Fluids* 92.12 (2020), pp. 1845–1865. doi: https://doi.org/10.1002/fld.4851. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.4851. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4851.

[145] J. M. Stokes et al. “A Deep Learning Approach to Antibiotic Discovery”. In: *Cell* 180.4 (2020), 688–702.e13. issn: 0092-8674. doi: https://doi.org/10.1016/j.cell.2020.01.021. url: https://www.sciencedirect.com/science/article/pii/S0092867420301021.

[146] P. Toth et al. “Hamiltonian Generative Networks”. In: *International Conference on Learning Representations*. 2020. url: https://openreview.net/forum?id=HJenn6VFvB.

[147] K. Um et al. “Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 6111–6122.

[148] Q. Wang, N. Ripamonti, and J. S. Hesthaven. “Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism”. In: *Journal of Computational Physics* 410 (2020), p. 109402.

[149] P. Wu et al. “Data-driven reduced order model with temporal convolutional neural network”. In: *Computer Methods in Applied Mechanics and Engineering* 360 (2020), p. 112766. issn: 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.112766.

[150] J. Zhuang et al. “Adaptive Checkpoint Adjoint Method for Gradient Estimation in Neural ODE”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 11639–11649.

[151] S. E. Ahmed et al. “On closures for reduced order models—A spectrum of first-principle to machine-learned avenues”. In: *Physics of Fluids* 33.9 (2021), p. 091301.

[152] E. de Bézenac, I. Ayed, and P. Gallinari. “CycleGAN Through the Lens of (Dynamical) Optimal Transport”. In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Ed. by N. Oliver et al. Cham: Springer International Publishing, 2021, pp. 132–147. isbn: 978-3-030-86520-7.

[153] S. L. Brunton et al. “Modern Koopman theory for dynamical systems”. In: *arXiv preprint arXiv:2102.12086* (2021).

[154] S. R. Bukka et al. “Assessment of unsteady flow predictions using hybrid deep learning based reduced-order models”. In: *Physics of Fluids* 33.1 (2021), p. 013601. doi: 10.1063/5.0030137.

[155] J. L. Callaham, S. L. Brunton, and J.-C. Loiseau. “On the role of nonlinear correlations in reduced-order modeling”. In: *arXiv preprint arXiv:2106.02409* (2021).

[156] N. Deng et al. “Route to Chaos in the Fluidic Pinball”. In: (Apr. 2021).

[157] S. Fresca and A. Manzoni. “Real-Time Simulation of Parameter-Dependent Fluid Flows through Deep Learning-Based Reduced Order Models”. In: *Fluids* 6.7 (2021). issn: 2311-5521. doi: 10.3390/fluids6070259.

[158] W. Gilpin. *Chaos as an interpretable benchmark for forecasting and data-driven modelling*. 2021. arXiv: 2110.05266 [cs.LG].

[159] S. Jain and G. Haller. “How to compute invariant manifolds and their reduced dynamics in high-dimensional finite element models”. In: *Nonlinear Dynamics* 107.2 (Oct. 2021), pp. 1417–1450. issn: 1573-269X. doi: 10.1007/s11071-021-06957-4. url: http://dx.doi.org/10.1007/s11071-021-06957-4.

[160] K. K. Lin and F. Lu. “Data-driven model reduction, Wiener projections, and the Koopman-Mori-Zwanzig formalism”. In: *Journal of Computational Physics* 424 (Jan. 2021), p. 109864. issn: 0021-9991. doi: 10.1016/j.jcp.2020.109864.

[161] Y. T. Lin et al. “Data-driven learning for the Mori-Zwanzig formalism: a generalization of the Koopman learning framework”. In: (2021).

[162] L. Lu et al. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229. doi: 10.1038/s42256-021-00302-5. url: https://doi.org/10.1038%2Fs42256-021-00302-5.

[163] L. Lu et al. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature machine intelligence* 3.3 (2021), pp. 218–229.

[164] R. Maulik, B. Lusch, and P. Balaprakash. “Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders”. In: *Physics of Fluids* 33.3 (2021), p. 037106. doi: 10.1063/5.0039986.

[165] C. Michelén Ströfer and H. Xiao. "End-to-end differentiable learning of turbulence models from indirect observations". In: (Apr. 2021).

[166] T. Pfaff et al. "Learning Mesh-Based Simulation with Graph Networks". In: *International Conference on Learning Representations*. 2021.

[167] A. Ramesh et al. "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[168] J. Brandstetter et al. *Geometric and Physical Quantities Improve E(3) Equivariant Message Passing*. 2022. arXiv: 2110.02905 [cs.LG].

[169] S. L. Brunton et al. "Modern Koopman Theory for Dynamical Systems". In: *SIAM Review* (2022).

[170] M. Bucci et al. "Nonlinear Optimal Control Using Deep Reinforcement Learning". In: Jan. 2022, pp. 279–290. isbn: 978-3-030-67901-9. doi: 10.1007/978-3-030-67902-6_24.

[171] P. Y. Chen et al. "CROM: Continuous reduced-order modeling of PDEs using implicit neural representations". In: *arXiv preprint arXiv:2206.02607* (2022).

[172] T. Dockhorn, A. Vahdat, and K. Kreis. "Score-Based Generative Modeling with Critically-Damped Langevin Diffusion". In: *International Conference on Learning Representations*. 2022. url: https://openreview.net/forum?id=CzceR82CYc.

[173] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. 2022. arXiv: 2109.14545 [cs.LG].

[174] C. Herrero Martin et al. "EP-PINNs: Cardiac Electrophysiology Characterisation Using Physics-Informed Neural Networks". In: *ront Cardiovasc Med.* (2022). doi: 10.3389/fcvm.2021.768419.

[175] P. Kidger. *On Neural Differential Equations*. 2022. doi: 10.48550/ARXIV.2202.02435. url: https://arxiv.org/abs/2202.02435.

[176] E. Menier et al. *Continuous Methods : Adaptively intrusive reduced order model closure*. 2022. arXiv: 2211.16999 [cs.LG].

[177] E. Menier et al. *Continuous Methods : Hamiltonian Domain Translation*. 2022. arXiv: 2207.03843 [cs.CV].

[178] T. N. K. Nguyen et al. “Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: An application to rubber calendering process”. In: *Engineering Applications of Artificial Intelligence* 114 (2022), p. 105176. doi: 10.1016/j.engappai.2022.105176. url: https://doi.org/10.1016%2Fj.engappai.2022.105176.

[179] A. Racca, N. A. K. Doan, and L. Magri. *Modelling spatiotemporal turbulent dynamics with the convolutional autoencoder echo state network*. 2022. arXiv: 2211.11379 [physics.flu-dyn].

[180] A. Racca and L. Magri. *Statistical prediction of extreme events from small datasets*. 2022. arXiv: 2201.08294 [physics.flu-dyn].

[181] J. Richter-Powell, Y. Lipman, and R. T. Q. Chen. “Neural Conservation Laws: A Divergence-Free Perspective”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh et al. 2022. url: https://openreview.net/forum?id=prQkA_NjuuB.

[182] R. Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[183] C. Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. doi: 10.48550/ARXIV.2205.11487. url: https://arxiv.org/abs/2205.11487.

[184] P. R. Vlachas et al. “Multiscale simulations of complex systems by learning their effective dynamics”. In: *Nature Machine Intelligence* (2022). doi: 10.1038/s42256-022-00464-w.

[185] E. Farzamnik et al. “From snapshots to manifolds – a tale of shear flows”. In: *Journal of Fluid Mechanics* 955 (2023), A34. doi: 10.1017/jfm.2022.1039.

[186] R. Geelen, S. Wright, and K. Willcox. “Operator inference for non-intrusive model reduction with quadratic manifolds”. In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115717. issn: 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115717. url: https://www.sciencedirect.com/science/article/pii/S0045782522006727.

[187] S. Kaltenbach, P.-S. Koutsourelakis, and P. Koumoutsakos. “Interpretable reduced-order modeling with time-scale separation”. In: *arXiv preprint arXiv:2303.02189* (2023).

[188] S. Kaltenbach, P. Perdikaris, and P.-S. Koutsourelakis. “Semi-supervised invertible neural operators for Bayesian inverse problems”. In: *Computational Mechanics* (2023), pp. 1–20.

[189] I. Kičić et al. *Adaptive learning of effective dynamics: Adaptive real-time, online modeling for complex systems*. 2023. arXiv: 2304.01732 [physics.comp-ph].

[190] Y. Liu et al. *Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models*. 2023. arXiv: 2304.01852 [cs.CL].

[191] E. Menier et al. "CD-ROM: Complemented Deep - Reduced order model". In: *Computer Methods in Applied Mechanics and Engineering* 410 (2023), p. 115985. issn: 0045-7825. doi: https://doi.org/10.1016/j.cma.2023.115985. url: https://www.sciencedirect.com/science/article/pii/S0045782523001081.

[192] E. Menier et al. *Interpretable learning of effective dynamics for multiscale systems*. 2023. arXiv: 2309.05812 [stat.ML].

[193] T. Nguyen et al. *ClimaX: A foundation model for weather and climate*. 2023. arXiv: 2301.10343 [cs.LG].

[194] R. Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[195] H. Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[196] N. Yokoyama et al. *Adaptive Skill Coordination for Robotic Mobile Manipulation*. 2023. arXiv: 2304.00410 [cs.RO].

[197] A. Kalur et al. "Data-driven closures for the dynamic mode decomposition using quadratic manifolds". In: *AIAA AVIATION 2023 Forum*. doi: 10.2514/6.2023-4352. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2023-4352. url: https://arc.aiaa.org/doi/abs/10.2514/6.2023-4352.

[198] Y. LeCun, C. Cortes, and C. J. C. Burges. *The MNIST database of handwritten digits.* url: http://yann.lecun.com/exdb/mnist/..

[199] Z. Liu et al. "Deep learning face attributes in the wild." In: *Proceedings of International Conference on Computer Vision (ICCV)*.

[200] S. Popinet. *Basilisk flow solver and PDE library*.

[201] P. Vlachas et al. "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks". In: *Proceedings of the Royal Society A* 474 (2213). doi: https://doi.org/10.1098/rspa.2017.0844.