N° d'ordre NNT : 2024ISAL0001

Thèse de Doctorat de l'INSA LYON,
membre de l'Université de Lyon

École doctoral 512
Informatique et Mathématique de Lyon (infomaths)

Spécialité de doctorat :
Informatique

Soutenue publiquement le 16 Janvier 2024, par :
Steeven Janny

Identification and Simulation of Physical Systems with Structured Deep Learning and Inductive Knowledge

---

Devant le jury composé de :

Gallinari	Patrick	Professeur des Universités, Univ. Sorbonne, Paris, France	Rapporteur
Di Meglio	Florent	Maître de Conférence, HDR Ecole des Mines de Paris, France	Rapporteur
Mansard	Nicolas	Directeur de Recherche CNRS, Univ. Paul Sabatier, Toulouse, France	Examinateur
Habrard	Amaury	Professeur des Universités, Univ. Jean Monnet, Toulouse, France	Examinateur
Thuerey	Nils	Professor, Technical University of Munich, Munich, Germany	Examinateur
Digne	Julie	Directrice de Recherche CNRS, INSA Lyon, Villeurbanne, France	Co-Directrice de thèse
Nadri	Madiha	Maître de Conférence, Univ. Claude Bernard, Villeurbanne, France	Co-Directrice de thèse
Wolf	Christian	Principal Scientist, NaverLabs Europe NaverLabs Europe, Meylan, France	Co-Directeur de thèse

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Steeven Janny, *Identification and Simulation of Physical Systems with Structured Deep Learning and Inductive Knowledge*, ©2023

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Département FEDORA – INSA Lyon - Ecoles Doctorales

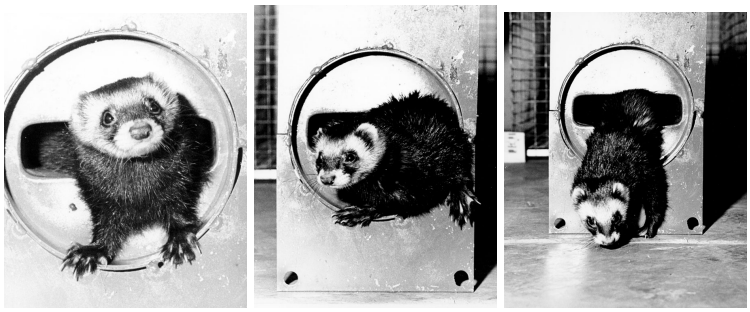
SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **https://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **https://edeea.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 **philippe.delachartre@insa-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**Mme Sandrine CHARLES** Université Claude Bernard Lyon 1 UFR Biosciences Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69622 Villeurbanne CEDEX **sandrine.charles@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://ediss.universite-lyon.fr** Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 **stephane.benayoun@ec-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **https://edsciencessociales.universite-lyon.fr** Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Bruno MILLY** Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 **bruno.milly@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Felicia coming out of a particle accelerator.



figures from history.fnal.gov

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Abstract

Recent technological progress is supported by the generalization of numerical tools for simulating, controlling, and observing physical systems. Yet, by focusing on more and more complex phenomena, our conventional tools are falling short of meeting the growing expectations of engineers, whether in terms of accuracy or computation time.

Data-driven approaches, in particular neural networks, offer promising alternatives to address these new challenges. These models can capture complex, nonlinear relationships in physical systems, and shift the burden from manual derivation of tedious mathematical formulas towards large-scale data collection. However, these methods often sacrifice stability, robustness, precision, and more generally guarantees classically offered by traditional approaches.

In this thesis, we propose combining the fields of physics, deep learning, and control theory to propose new hybrid methods, taking advantage of the expressivity of neural networks, while relying on inductive biases from physics. We describe theoretical tools (discussed in Part 1) related to the simulation of dynamical systems and connect them to neural network design. In a second time (Part 2), we leverage these insights to design control algorithms and simulation techniques addressing the resolution of complex problems related to partial differential equations. Finally, in Part 3, we focus on larger-scale simulations such as fluid dynamics and counterfactual reasoning.

Our work has been presented at scientific conferences in the field of artificial intelligence and control theory. By bridging the gap between physics and machine learning, we believe that this paves the way toward a new generation of methods for the simulation and control of physical systems.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Résumé

Les progrès technologiques de notre époque sont soutenus par la disponibilité croissante d'outils numériques pour simuler, contrôler et observer les systèmes physiques. En se concentrant sur des phénomènes de plus en plus complexes, nos outils conventionnels ne parviennent pas à répondre aux attentes croissantes des ingénieurs, que ce soit en termes de précision ou de temps de calcul.

Les approches data-driven, en particulier les réseaux de neurones, offrent des alternatives prometteuses pour résoudre ces problèmes. Ces types de modèles capturent des relations complexes et non linéaires dans les systèmes physiques et déplacent la charge de modélisation vers celle de la collecte de données. Cependant, ces nouvelles méthodes sacrifient souvent les critères de stabilité, de robustesse et de précision et plus généralement les garanties offertes par les approches traditionnelles.

Nous proposons de combiner les domaines de la physique, de l'apprentissage profond et de la théorie du contrôle pour proposer de nouvelles méthodes hybrides, tirant parti de la puissance des réseaux de neurones, tout en s'appuyant sur des biais inductifs issus de la physique. Ce manuscrit présente nos travaux dans ce domaine. En particulier, il décrit des outils théoriques (abordés dans la partie 1) liés à la simulation de systèmes dynamiques et les connecte à la conception de réseaux neuronaux. Dans un deuxième temps (Partie 2), nous exploitons ces connaissances pour concevoir des algorithmes de contrôle et des techniques de simulation impliquant la résolution de problèmes complexes liés aux équations aux dérivées partielles. Enfin, dans la troisième partie, nous abordons des problèmes de simulation à plus grande échelle tels que la dynamique des fluides et le raisonnement contrefactuel.

Nos travaux ont été présentés lors de conférences scientifiques dans le domaine de l'intelligence artificielle et de la théorie du contrôle. En construisant un pont entre la physique et l'apprentissage automatique, nous croyons fermement que cette direction de recherche peut contribuer à une nouvelle génération de méthodologies pour la simulation et le contrôle des systèmes physiques.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Acknowledgements

Acknowledgements will be added here after the defense

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Contents

Abstract i

Résumé iii

Acknowledgements v

Contents vii

List of Figures xi

List of Tables xv

Acronyms xvii

1 Introduction 1

1.1 A New Hope: Data-driven models for physics and control 1

1.2 The Physics Menace: Limitations of physics for models and simulations 2

1.3 Control Theory Strikes Back: Theoretical tools to interact with real world 4

1.4 The Return of Deep Learning: Hybrid approach to physics problems 7

1.5 Attack of the Ph.D.: organization of the manuscript 8

I State of the art 10

General remarks 11

2 Neural networks for system identification 13

2.1 The Lord of the Physics: an introduction to dynamical systems 13

2.2 One Training Method to Rule Them All 19

2.3 The Fellowship of the Dynamical Systems 23

2.4 The Two Approaches: hybrid models 28

2.5 The Return of Physics 33

2.6 Take-home messages 34

3 Deep Learning for intuitive physics 35

3.1 Learning the solution from PDE operator 36

3.2 Learning the solution from sparse observations 39

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

3.3 Learning the solver for grid-based data . . . 44
3.4 Learning the solver for mesh-based simulations . . . 51
3.5 Large scale datasets for physics . . . 54
3.6 Take-home messages . . . 55

II Foundations for Robust Simulations using Observer Theory 58

General remarks **59**

4 Learning Reduced Nonlinear State-Space Models **61**
4.1 Context . . . 61
4.2 Problem statement and preliminary results . . . 63
4.3 Modeling and learning . . . 66
4.4 Experimental results . . . 69
4.5 Conclusion . . . 74
4.6 Post-scriptum: taking a step back . . . 74

5 Deep-KKL **79**
5.1 Context . . . 79
5.2 Prediction via embedding into an output-dependent uniform contraction . . . 81
5.3 A possible solution via KKL . . . 84
5.4 Learning KKL with deep networks . . . 86
5.5 Numerical simulations . . . 88
5.6 Conclusion . . . 91
5.7 Post-scriptum: taking a step back . . . 92

III Differential equations for simulation and control 96

General remarks **97**

6 Output tracking via contraction theory **99**
6.1 Context . . . 99
6.2 An introduction to contraction theory . . . 100
6.3 Preliminaries . . . 103
6.4 Main results . . . 104
6.5 Simulations . . . 110
6.6 Conclusions . . . 112
6.7 Post-scriptum: taking a step back . . . 112

7 Space and time continuous simulation **115**
7.1 Context . . . 115
7.2 Continuous solutions from sparse observations . . . 117
7.3 Experimental results . . . 123

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

7.4 Conclusion . . . 126
7.5 Post-scriptum: taking a step back . . . 127

IV Scaling up: large-scale learning of complex physics phenomena 130

General remarks **131**

8 Turbulent fluid dynamics with mesh transformers **133**
8.1 Context . . . 133
8.2 The Eagle dataset and benchmark . . . 135
8.3 Learning unsteady airflow . . . 137
8.4 Experiments . . . 140
8.5 Conclusion . . . 144
8.6 Post-scriptum: taking a step back . . . 145

9 Filtered-CoPhy **147**
9.1 Context . . . 147
9.2 The Filtered-CoPhy benchmark . . . 149
9.3 Unsupervised learning of counterfactual physics . . . 152
9.4 Experiments . . . 156
9.5 Conclusion . . . 159
9.6 Post-Scriptum: taking a step back . . . 159

10 Final remarks **163**
10.1 Theoretical insights for more principled models . . . 163
10.2 Physics and deep learning for robotics . . . 164
10.3 Neural simulators for faster engineering . . . 164

Bibliography **167**

V Appendices 193

A Appendix of chapter 4 **195**
A.1 Proof of proposition 4.2 . . . 195
A.2 Model details . . . 195
A.3 Dataset details . . . 196

B Appendix of chapter 5 **197**
B.1 Proof of Proposition 5.1 . . . 197
B.2 Proof of Theorem 5.1 . . . 197
B.3 Proof of Proposition 5.2 . . . 198
B.4 Proof of Proposition 5.3 . . . 199

C Appendix of chapter 6 **201**
C.1 Proof of Proposition 6.1 . . . 201

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

C.2 Proof of Proposition 6.2 . . . 202
C.3 Model details . . . 203

D Appendix of chapter 7 **205**
D.1 Proof of proposition 7.1 . . . 205
D.2 Comparison of upper bounds in Proposition 7.1 . . . 206
D.3 Proof of proposition 7.2 . . . 207
D.4 Model description . . . 208
D.5 Baselines and datasets details . . . 210
D.6 More results . . . 212

E Appendix for chapter 8 **219**
E.1 Dataset details . . . 219
E.2 Model details . . . 222
E.3 More results . . . 224

F Appendix for chapter 9 **233**
F.1 Further details on dataset generation . . . 233
F.2 Performance evaluation of the de-rendering module . . . 237
F.3 Comparison with the Transporter baseline . . . 239
F.4 Details of model architectures . . . 242
F.5 Additional quantitative evaluation . . . 246
F.6 Experiments on real-world data . . . 247
F.7 Qualitative evaluation: more visual examples . . . 248

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

List of Figures

1.1 Introduction: Sand simulation . . . 3
1.2 Introduction: Qualitative comparison of turbulence models . . . 4
1.3 Introduction: Block diagram of control theory . . . 5
1.4 Introduction: Overviews of publications during the Ph.D. . . . 9

2.1 SOTA: Lorenz Attractor . . . 14
2.2 SOTA: Latent Dynamics . . . 18
2.3 SOTA: Neural ODE . . . 20
2.4 SOTA: Finite Elements Networks . . . 23
2.5 SOTA: Deep Koopman operator . . . 25
2.6 SOTA: Hybrid dynamics . . . 30
2.7 SOTA: Physics-guided training . . . 32

3.1 SOTA: Physics-Informed Neural Networks . . . 37
3.2 SOTA: Competitive PINN . . . 39
3.3 SOTA: DeepONet . . . 41
3.4 SOTA: DINo . . . 43
3.5 SOTA: DeepFluid . . . 46
3.6 SOTA: PhyDNet . . . 47
3.7 SOTA: Object-level visual reasonning . . . 50
3.8 SOTA: Encode-Process-Decode . . . 53

4.1 Canonical state-space: Model overview and training pipeline . . . 67
4.2 Canonical state-space: Qualitative results . . . 70
4.3 Canonical state-space: Comparison with the GRU baseline . . . 71
4.4 Canonical state-space: Ablation study on compression . . . 74

5.1 Deep-KKL: Model overview . . . 82
5.2 Deep-KKL: Qualitative results . . . 88
5.3 Deep-KKL: Noise robustness . . . 89
5.4 Deep-KKL: Generalization to OOD . . . 90
5.5 Deep-KKL: Evaluation on noisy setup . . . 93

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

6.1 Contractive control: Model overview . . . 107
6.2 Contractive control: Steady state generator . . . 110
6.3 Contractive control: Qualitative results on output tracking . . . 111

7.1 Continuous solution: Model overview . . . 117
7.2 Continuous solution: Model implementation . . . 122
7.3 Continuous simulation: Qualitative results on Eagle . . . 126

8.1 Eagle: Teaser . . . 134
8.2 Eagle: Snapshots from the dataset . . . 136
8.3 Eagle: Model overview . . . 138
8.4 Eagle: Qualitative results . . . 141
8.5 Eagle: Locality of reasoning . . . 143
8.6 Eagle: Impact of cluster size . . . 144

9.1 Filtered-CoPhy: Dataset overview . . . 150
9.2 Filtered-CoPhy: Impact of temporal frequency . . . 151
9.3 Filtered-CoPhy: Derendering module overview . . . 152
9.4 Filtered-CoPhy: Model overview . . . 156
9.5 Filtered-CoPhy: Qualitative results . . . 158

D.1 Continuous solution: Caveat MaGNeT . . . 211
D.2 Continuous solution: Quantitative results on Shallow-Water . . . 213
D.3 Continuous solution: Qualitative results for time continuity . . . 214
D.4 Continuous solution: Ablations . . . 216
D.5 Continuous solution: Attention maps . . . 217
D.6 Continuous solution: Parameter study . . . 218

E.1 Eagle: Down-sampling from raw simulation . . . 221
E.2 Eagle: Mesh to grid conversion . . . 222
E.3 Eagle: K-number maps . . . 225
E.4 Eagle: Detailed metrics . . . 226
E.5 Eagle: Example on Cylinder Flow . . . 227
E.6 Eagle: Example on Scalar-Flow . . . 228
E.7 Eagle: Example on Eagle . . . 229
E.8 Eagle: Failure cases . . . 230

F.1 Filtered-CoPhy: Threshold parameter . . . 235
F.2 Filtered-Cophy: Sanity check . . . 235
F.3 Filtered-CoPhy: Dataset balance . . . 236
F.4 Filtered-CoPhy: Reconstruction task . . . 238
F.5 Filtered-CoPhy: Coefficients sweep . . . 239

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

F.6 Filtered-CoPhy: Comparison with Transporter 240
F.7 Filtered-CoPhy: Static keypoints 242
F.8 Filtered-CoPhy: Keypoints consistency 243
F.9 Filtered-CoPhy: Effect of the do-operation 248
F.10 Filtered-CoPhy: Real-world dataset 249
F.11 Filtered-CoPhy: Examples from BlockTower-CF 250
F.12 Filtered-CoPhy: Examples from Balls-CF 251
F.13 Filtered-CoPhy: Examples from Collision-CF 252

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

List of Tables

4.1 Canonical state-space: Quantitative results . . . 72

5.1 Deep-KKL: Description of system used for evaluation . . . 87
5.2 Deep-KKL: Quantitative results . . . 88
5.3 Deep-KKL: Different state size . . . 92

6.1 Contractive control: Quantitative results . . . 111

7.1 Continuous simulation: Space continuity . . . 125
7.2 Continuous simulation: Time continuity . . . 127

8.1 Eagle: Comparison with existing datasets . . . 135
8.2 Eagle: Quantitative results . . . 141
8.3 Eagle: Ablations . . . 142
8.4 Eagle: Generalization to unseen geometries . . . 144

9.1 Filtered-CoPhy: Quantitative results . . . 156
9.2 Filtered-CoPhy: Comparison with copying baselines . . . 156
9.3 Filtered-CoPhy: Ablation on coefficients . . . 157
9.4 Filtered-Cophy: Ablation on CoDy . . . 159
9.5 Filtered-CoPhy: Ablation of the filters . . . 159

D.1 Continuous solution: Time extrapolation . . . 212
D.2 Continuous solution: Generalization to unseen grid . . . 215
D.3 Continuous solution: Ablation on interpolation module . . . 215

E.1 Eagle: Mesh-downsampling . . . 231

F.1 Filtered-CoPhy: Sanity check . . . 235
F.2 Filtered-CoPhy: Reconstruction task . . . 237
F.3 Filtered-CoPhy: Number of coefficients . . . 238
F.4 Filtered-CoPhy: Comparison with Transporter . . . 240
F.5 Filtered-CoPhy: Model details . . . 244
F.6 Filtered-CoPhy: MOT metrics . . . 247

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Acronyms

AI	Artificial Intelligence
CF	Counterfactual
CNN	Convolutional Neural Network
FEM	Finite Element Method
FVM	Finite Volume Method
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
INR	Implicit Neural Representation
KKL	Kazantzis-Kravaris/Luenberger
LES	Large Eddies Simulation
LSTM	Long-Short Term Memory
MLP	Multi-Layer Perceptron
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
RANS	Reynolds-Averaged Navier-Stokes
RK	Runge-Kutta
RL	Reinforcement Learning
RNN	Recurrent Neural Network
sim2real	Simulation to Real
UAV	Unmanned Aerial Vehicle

Throughout the manuscript, we used three symbols to indicate:

An example or a remark.

[Paper name] A work we want to highlight and explain in more detail.

Proof: A mathematical proof. ■

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 1

Introduction

1.1 A New Hope: Data-driven models for physics and control

According to Ernest Rutherford, science is either physics or stamp collecting. This provocative maxim (Rutherford won a Nobel prize in chemistry) emphasizes the tremendously difficult purpose of physics: explaining the behavior of the world through mathematics. Supported by centuries of research and discoveries, the field has achieved significant successes, providing fundamental laws explaining most everyday phenomena. Today, it ventures into exploring profound concepts such as the origin of the universe and the nature of matter.

Physics plays a prominent role in many fields, aiding in developing tools to not only understand but also interact with the world. By collaborating with control theory and mathematics, it provides methods to build controllers for regulating the temperature of supercomputers, design optimized airplane profiles, and accelerate vaccine discovery. Yet, as physicists delve into more and more advanced problems, explicit approaches for physical modeling and control start to hit their limitations. The trade-off between model precision and complexity hinders engineers from effectively mastering new phenomena, such as plasma stabilization in nuclear fusion, or meeting the growing expectations for fast and accurate fluid simulations. Nowadays, engineers require new, sophisticated tools that overcome the limitations of conventional physics and control methods.

In contrast, modern data-driven approaches, in particular neural networks, offer a fresh perspective to automatically interpret physics by learning from large-scale datasets. They may discover complex interactions neglected by handcrafted models, learn shortcuts to accelerate computationally expensive simulations, or predict inputs to drive a system toward a chosen configuration. However, deep learning is still in its early stages and has primarily focused on tasks such as computer vision and natural language processing, which, although complex, have different requirements from physics and control problems.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Thus, the situation is the following: on one side of the spectrum stand engineers and physicists seeking new ways to interact with the world, while being limited by conventional tools. On the other side, deep learning provides powerful tools to extract complex behavior from large datasets, but it falls short of meeting industrial-grade requirements, such as long-horizon accuracy in physics simulation and stability guarantees for controllers.

Our goal is to bridge the gap between both domains. We propose to draw inspiration from physics and control tools to guide deep learning-based models toward more effectively addressing physics-related problems. Throughout the manuscript, we distill insights from control theory to enhance the ability of neural networks to tackle tasks related to dynamical systems. Our goal is to explore the spectrum between traditional and data-driven methods by introducing (a) new architectures of neural networks for physics simulation and control, rooted in prior knowledge from both fields, and (b) large-scale datasets tailored for deep learning of physics-oriented tasks that will benefit the community.

1.2 The Physics Menace: Limitations of physics for models and simulations

Physics has provided a comprehensive mathematical framework to describe the phenomena surrounding us. Certainly, captivating research areas and recent discoveries persist within the domain, e.g. quantum gravity or fundamental particles. Nonetheless, we currently possess a rich set of physics equations at our disposal, empowering engineers to simulate the behavior of the world. For instance:

🔍 Most of the components of a high-speed train (TGV) are derived from simulations: the shape of the locomotive minimizes friction according to the Navier-Stokes equations, the brakes efficiently cool down thanks to the heat equation, and the design of joints between cars results of a simulation problem. The rail network is also guided by signals emitted as waves, obeying the Maxwell equations.

Yet, despite our knowledge of natural processes, we are facing more and more challenges in understanding increasingly complex physics systems. These difficulties mainly stem from technical and/or human limitations, in particular when physics phenomena are beyond what can reasonably be addressed with a human brain. Such situations may arise from three different sources:

1. **Incomplete modeling of physics** results from an oversimplification of the world. Designing a model requires neglecting or approximating certain phenomena, either because they are too complex, or because we believe they are insignificant. This necessarily leads to Simulation to Real (sim2real) gaps.

   🔍 Modeling the dynamics of granular materials is extremely difficult due to the enormous number of particles involved. Thus, computing analytic equations of the interac-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

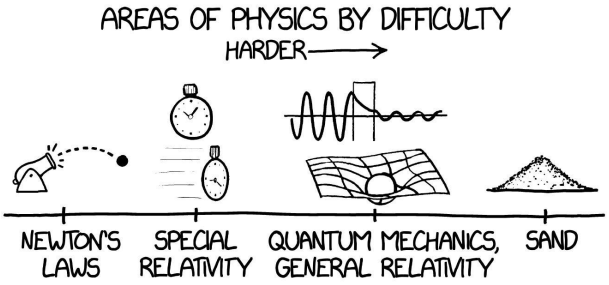


Figure 1.1: **Sand simulations** – are surprisingly difficult to model due to the enormous number of interactions to account for when simulating the system (drawing from the New York Times, Randall Munroe).

tions by hand is merely impossible, as we are limited by the complexity we can handle. Behaviors may also be neglected voluntarily to simplify the model, e.g. most kinematic models in robotics neglect plays in the joints.

2. **Numerical errors and instabilities** may appear when discretizing and numerically solving equations, and arise independently from the quality of the model. The transition from a continuous equation to a discrete solution is not trivial: algorithms must be carefully designed to maintain physical properties while preventing numerical instabilities. Moreover, the success of a simulation heavily depends on the spatial and temporal resolution to limit approximation errors.

   Fluid mechanics is an infamous example of challenging simulations, due to the failure of simple numerical solvers to maintain mass conservation in the discrete solution. More advanced numerical solvers with smaller discretization schemes are necessary to obtain good results.

3. **Computational limitations** often impede accurate simulations due to the inherent complexity and scale of the problem at hand. Even with an accurate model and a trustworthy numerical solver, the computation power required to solve the task within an acceptable delay may not always be available.

   Model Predictive Control (MPC) is a popular control algorithm leveraging a dynamical model to estimate optimal inputs for tracking a user-defined trajectory. However, it involves a demanding non-convex optimization process, thus advocating for simple models and fast algorithms for real-time and on-board implementation.

We illustrate some of these challenges in figure 1.2, where we conduct a simulation of a 2D air flow onto a cone. We used the same spatial and temporal discretization scheme for all simulations, yet with different physics models: from the simplest Reynolds-Averaged Navier-Stokes (RANS) with Spalart–Allmaras equation for turbulence, to the most accurate Large Eddies Simulation (LES). The precision of the simulations evolves with the accuracy of

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

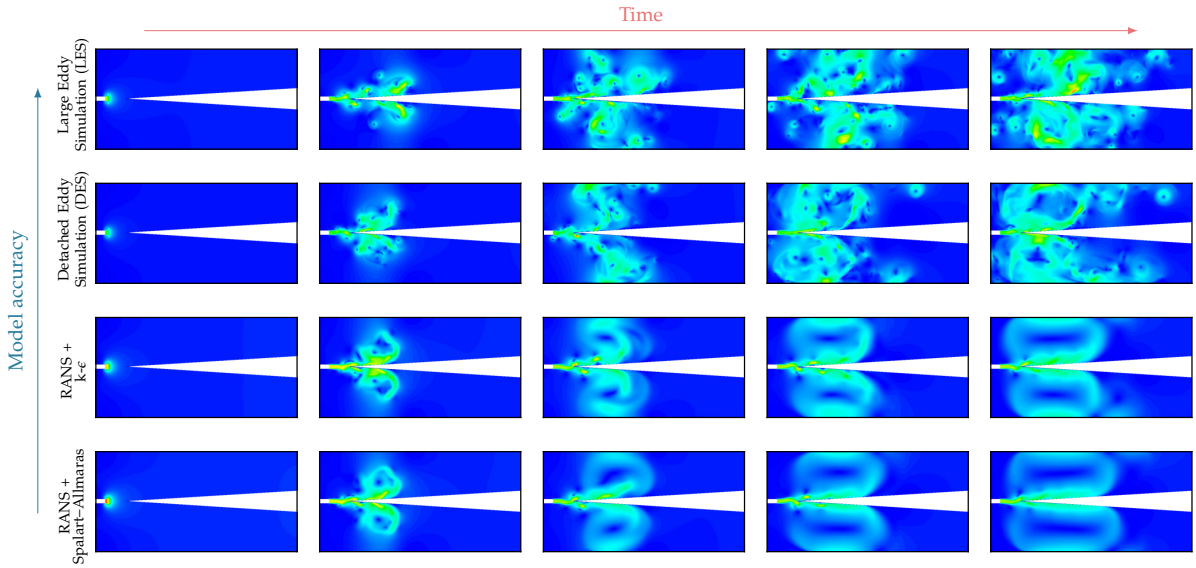


Figure 1.2: **Qualitative comparison of turbulences models** – in these 2D simulations, the air is expelled from a nozzle onto a cone at $10\,\mathrm{m\,s^{-1}}$. We used the same spatial and temporal discretization but changed the turbulence model. The RANS equations are augmented with two different turbulence models: Spalart-Allmaras (one equation) or $k-\epsilon$ (two equations) and produce relatively smooth airflows. In contrast, the more accurate LES model exhibits additional vortices and a significantly more intricate flow. However, this result comes at the cost of increased computation time: the RANS simulations were completed in approximately 30 minutes, whereas the LES on the same machine required about one hour

the models. However, accuracy comes at the price of computation time: the LES experiment required twice as much time as the one using RANS and Spalart-Allmaras.

Consequently, merely understanding the laws of physics is not enough to achieve accurate simulations. Engineers must grapple with errors, approximations, and the risk of instabilities. To be able to interact with the real world despite these limitations, control theory proposes to leverage mathematical analysis of the behavior of physical systems to not only address model uncertainties but also to encompass efficient command policies and measurement techniques.

1.3 Control Theory Strikes Back: Theoretical tools to interact with real world

If you had the chance to steer a modern commercial drone[1], you probably noticed how pleasant and simple they are to control for a beginner. Despite their amazing agility, drones remain stable, even during windy flights. Commands are responsive, with a relatively small learning curve, and some drones encompass vision-based features, allowing you to keep objects in sight while moving almost freely in the air.

All these features (and many others) are powered by control theory: a branch of engineer-

[1] otherwise, we highly recommend the experience.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

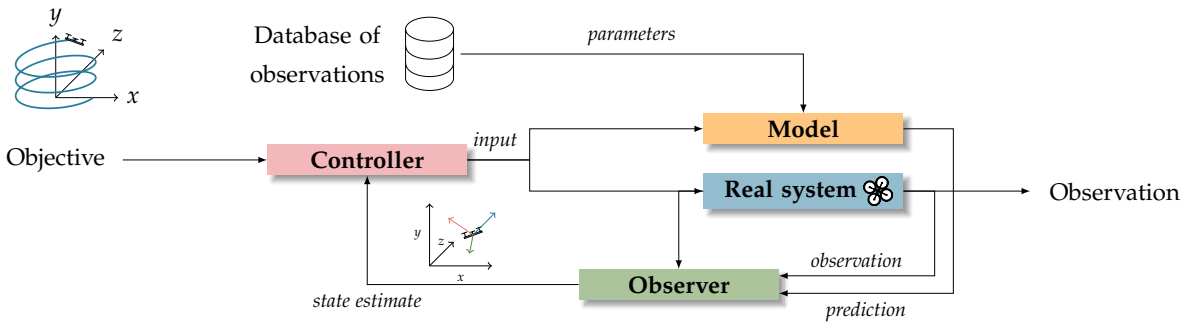


Figure 1.3: **Block diagram from Control Theory** – the controller computes the *inputs* for the real system in order to track some user-defined reference signal. This is achieved by considering a *state estimate* provided by an observer leveraging noisy *observations* as well as *predictions* from a model to approximate the complete state of the system. The *parameters* of the model are given by a system identification phase relying on a dataset of observations.

ing and mathematics dealing with the behavior of dynamical systems and aiming to regulate, monitor, and optimize their performance. This discipline has been deeply ingrained in various applications, from early cruise control systems in automobiles to sophisticated flight control mechanisms in modern aircraft. In essence, control theory enables us to design systems that can handle uncertainties and disturbances while operating efficiently and robustly. The area of research is very rich and active but could be roughly clustered into three categories, pictured in figure 1.3:

1. **System identification/modeling** consists of designing a (parametric) mathematical model of the temporal evolution of the physical system, which usually takes the form of an ordinary or partial differential equation. The model can be designed from physical considerations, or be purely parametric (hence, neural networks). The system identification step identifies the parameters of this model via a handful of measurements gathered from the system.

2. **Observers** retrieve and track the complete state of a system from partial and/or noisy measurements. Common techniques, such as Kalman filtering or the Luenberger observer (Kalman, 1960; Luenberger, 1971) leverage a dynamical model (gathered from the system identification step above) and fuse its prediction to the sensor outputs to achieve real-time state reconstruction.

3. **Controllers** produce inputs to drive the system towards some reference signals or ensure stability. A good controller must be robust to different sources of uncertainties (electronic, mechanical vibrations, etc.) and external perturbations (wind, etc.) while guaranteeing the stability of the system.

Despite the strengths of control theory in dealing with uncertainties, its reliance on mathematical analysis limits its practical usage. Industrial implementations often resort to simple linearized models and PID controllers instead of more advanced theoretical frameworks, lead-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

ing to potential performance gains left untapped. On the other hand, simpler algorithms hinder the exploration of complex dynamical systems that require advanced tools. Consequently, we face the following limitations:

- **Handling complex physics**: many control algorithms assume properties on the system of interest, such as stationarity, time-invariance, or linearity of dynamics, but real-world systems exhibit time-dependent and non-linear behaviors. Simple workarounds such as linearization provide limited approximations and may be very sensible to external perturbations.

  The dynamics of chemical reactions can be highly non-linear and time-dependent which greatly complicates the design of controllers. In practice, the behavior of the system changes drastically with temperature and pressure, limiting the domain of validity of a linearized model.

- **Complex implementation**: techniques addressing non-linear systems are complicated. The implementation of sophisticated control algorithms often involves analytic design and fine-tuning, which can be time-consuming and require expertise. The process may also involve challenging mathematical derivations to achieve satisfactory performance, especially for non-linear systems.

  Designing a robust control system for a humanoid robot capable of walking and navigating through different environments requires advanced control algorithms and parameter tuning to ensure stability and adaptability. Boston Dynamics© robots are good examples. They are driven by a distributed system, where each link is controlled by its own specifically designed controller. The complete robot requires a tremendous amount of work to design, implement, and tune each controller in order to plan its behavior with this hierarchical control algorithm.

- **High-Dimensional Data**: traditional control methods are best suited for small-scale systems with a limited number of variables. As the state-space dimensionality increases, computational complexity grows accordingly, making traditional control less practical for high-dimensional systems.

  Nuclear fusion requires stabilizing plasma into a precise shape using powerful magnetic fields which must be modulated dynamically. The physical state describing the system comprises parameters describing the plasma shape (in 3D) and physical quantities characterizing each magnetic field generator. Up to now, stabilization is approached using dozens of PID controllers which struggle to maintain stability after a couple of seconds.

Control theory has succeeded in overcoming the weaknesses of physics modeling, thus enabling interaction with the world, and is widely used in industrial applications. However,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

the increasing complexity of engineering tasks highlights the limitations of analytical methods, which motivates the use of data-driven techniques.

1.4 The Return of Deep Learning: Hybrid approach to physics problems

We have drawn an alarming picture so far: on one hand, physics simulations suffer from poor scalability to large-scale problems and struggle to retrieve physics laws in complex phenomena. In practice, the use of simulation is limited by computation time, while accuracy relies on our capacity to manually derive an accurate model of the system. On the other hand, the usability of control theory is limited due to mathematical obstacles and thus falls short of addressing new challenges.

The interest of these communities in the promises of deep learning is thus reasonable: neural networks offer impressive performance, competitive computation time, and the ability to extract reasoning patterns from data. Deep learning, control, and physics already met at the end of the 20th century, and many recent works find inspiration from older research dating from this period of limited computational power. This regain of interest may be explained by the shift of focus enabled by modern deep learning techniques. Eventually, the limitations of physics and control mentioned above derive from the same constraint:

For advanced problems, there are physical quantities, mathematical objects, and theoretical elements that we do ***not*** *want to compute nor model* ***by hand****.*

By focusing on data collection rather than complex modeling, data-driven methods shift the burden from manual derivation of tedious mathematical formulas toward large-scale data collection and let the algorithm extract necessary information on its own by detecting and exploiting regularities in the data. This approach is particularly beneficial when dealing with high-dimensional data or complex interactions. Neural networks, being numerically efficient, can also provide faster solutions compared to conventional methods.

However the interplay is reciprocal, and deep learning-based models have their weaknesses. They are known to generalize poorly to out-of-distribution inputs and are vulnerable to perturbations. Even within the training distribution, neural networks can occasionally exhibit singularities for specific inputs, and lead to erroneous predictions. Moreover, the community lacks theoretical results on confidence bounds. When used as auto-regressive models (e.g. to iteratively forecast the temporal evolution of a system), neural networks suffer from error accumulation and occasional instabilities, causing prediction errors to explode for certain inputs. We believe that, by incorporating principles from physics and control and leveraging structured reasoning and comprehensive analysis of dynamical systems, we can enhance the robustness, reliability, and performance of our deep learning-based models.

The cooperation between the fields of control, physics, and deep learning offers incredible

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

potential to address the complex challenges that engineers and researchers face today. Apart from the purely scientific interest of this field of research, it also benefits from strong interests from industry, which is already working on hybrid models for simulation, control, and observation of physical systems.

1.5 Attack of the Ph.D.: organization of the manuscript

Our work is driven by the *Ansatz* that merging deep learning with control theory and physics will lead to discoveries and improvements for each field. This combination presents exciting opportunities but also requires to clearly define the expectations and objectives for each project.

After a review of the state-of-the-art in Part I, we explore technically simpler yet fundamental questions in the realm of control theory. Part II of the manuscript discusses the properties that a data-driven model must possess to accurately simulate dynamical systems from a fundamental perspective. This part of our work addresses issues related to control theory and is thus focused on providing provably good performance or properties.

Related papers:

- Learning Reduced Nonlinear State-Space Models: an Output-Error Based Canonical Approach. **Steeven Janny**, Quentin Possamaï, Laurent Bako, Madiha Nadri, Christian Wolf. (2022). *In IEEE Conference on Decision and Control (CDC)*.

- Deep KKL: Data-Driven Output Prediction for Non-Linear Systems. **Steeven Janny**, Vincent Andrieu, Madiha Nadri, Christian Wolf, (2021). *In IEEE Conference on Decision and Control (CDC).*

In part III, we shift towards practical applications for physical simulation. Our focus is on resolving differential equations in a continuous manner, a requirement arising in many situations. In particular, we present a new hybrid control algorithm relying on deep learning for solving a very complex partial differential equation problem. We then expand our work to the general case of solving physics problems in a continuous manner using discrete observations.

Related papers:

- Deep Learning-Based Output Tracking via Regulation and Contraction Theory. Samuele Zoboli[2], **Steeven Janny**[2], Mattia Giaccagli[2]. (2023).*In International Federation of Automatic Control (IFAC) World Congress*

- Everything, Everywhere, All at Once: Continuous Solutions to PDEs from Sparse Observations. **Steeven Janny**, Julie Digne, Madiha Nadri, Christian Wolf. (2023). *Under review*

Finally, by focusing on the deep learning community, we identified a lack of large-scale

[2]equal contribution

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

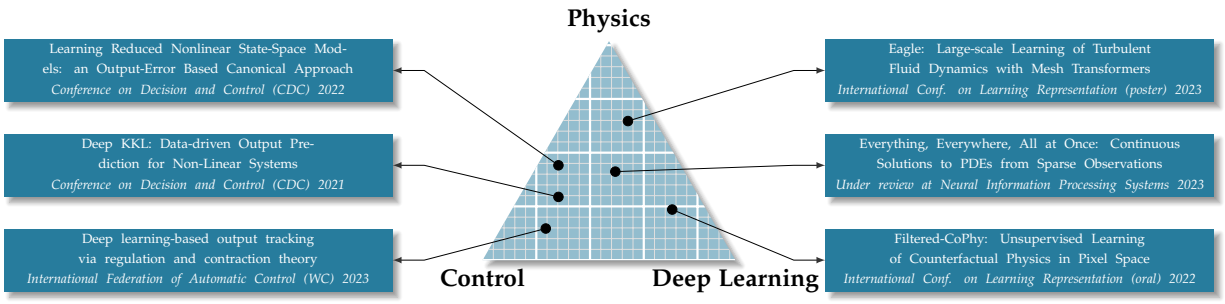


Figure 1.4: **Overviews of publications during the Ph.D.** – our work lies at the frontier of three research fields: physics, control theory, and deep learning. During this thesis, we have explored the spectrum of transversal approaches between domains, on one hand by bringing deep learning techniques to the automatic control community, and on the other hand by addressing physics simulation challenges via large-scale neural networks.

datasets and high-capacity models for simulating physical phenomena. To address this gap, we introduced a new fluid mechanics dataset and a novel, faster simulation structure. Additionally, we explored the concept of causality, crucial in physics, and introduced a new task, benchmark, and model for learning counterfactual physics. These contributions are presented in Part IV.

Related papers:

- Eagle: Large-Scale Learning of Turbulent Fluid Dynamics with Mesh Transformers. **Steeven Janny**, Aurélien Bénéteau, Madiha Nadri, Nicolas Thome, Julie Digne, Christian Wolf. (2023). *In International Conference on Learning Representation (ICLR), Poster.*

- Filtered-CoPhy: Unsupervised Learning of Counterfactual Physics in Pixel Space. **Steeven Janny**, Fabien Baradel, Natalia Neverova, Madiha Nadri, Greg Mori, Christian Wolf. (2022). *In International Conference on Learning Representation (ICLR), Oral.*

Across our work, we covered a wide spectrum of the field, collaborating with many fellow researchers coming from different fields. Consequently, researchers in deep learning might find the task addressed in the first parts of the manuscript relatively accessible, yet the automatic control community will be interested in the theoretical foundations required for integrating our models into their applications. As we progress, the tasks we tackle become more and more intricate and require high-capacity models trained on large-scale settings. These tools exhibit greater computational power, albeit at the expense of reduced analytical tractability. However, they enable us to achieve performance difficult to match with conventional approaches.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Part I

State of the art

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

General remarks

During the 1,074 days since the start of this PhD, the arXiv platform received 1,312 AI-related[3] submissions mentioning "*physics*" in the title, 1,014 mentioning "*simulation*", and 343 mentioning "*dynamical systems*" (with potential overlaps). To be more representative, one might also take into account submissions from physics mentioning AI ("*deep learning*": 1,249, "*neural network*": 2,187). The domain holds an exponentially growing amount of work that needs to be classified and clusterized for these bibliographic chapters. To address this task, we propose to distinguish two categories:

Chapter 2 : Neural networks for system identification

TLDR; This chapter discusses the identification of dynamical system using data-driven approaches. In particular, we focus on identifying general-purpose dynamical models that can be used for any downstream task, such as simulation or control. After a short introduction to dynamical systems, we present different techniques to (1) enforce properties to the learned dynamics, (2) train continuous dynamical models, (3) embed inductive priors, and (4) retrieve analytic equations using data.

Chapter 3 : Deep Learning for intuitive physics

TLDR; This chapter focuses on goal-oriented methods addressing intuitive physics with an emphasis on simulation tasks. Conversely to the previous chapter, we relax the requirement for an explicit dynamical model, which can simply be entangled in the neural network. These methods are less restricted in their design, and are tailored to achieve good performance on a given task. We discuss hybrid techniques for learning intuitive physics with different approaches depending on the type and amount of prior knowledge available for training.

[3] in cs.AI, cs.CV, cs.LG and cs.NE

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 2

Neural networks for system identification

2.1 The Lord of the Physics: an introduction to dynamical systems

2.1.1 *State-space representation*

We shall start this chapter following the tradition of research papers in control theory, and without further ado, consider a non-linear dynamical system of the form

$$\dot{\boldsymbol{s}}(t) = f\big(\boldsymbol{s}(t)\big), \quad \forall t \in \mathbb{R}^+. \tag{2-1}$$

This equation is called the *state-space representation* and models the dynamics of a physical system. It takes the form of an Ordinary Differential Equation (ODE) over a *state variable* $\boldsymbol{s}(t) \in \mathbb{R}^n$ depending on time t. To understand why ODEs appear in most physical systems, one might think about the conservation principle. Every closed system must admit physical quantities that remain constant, regardless of the transformations it undergoes. These conservation laws are central in physics, and state that energy, momentum, or charge (as well as exotic quantities from quantum mechanics) cannot vary in time. The evolution of a physical system can be described by studying the flow of these quantities as they transform from one form to another. An ODE fulfills this purpose by expressing the time derivative of the state $\dot{\boldsymbol{s}}$ as the results of flows encapsulated in the dynamic function, or simply *dynamics*, f.

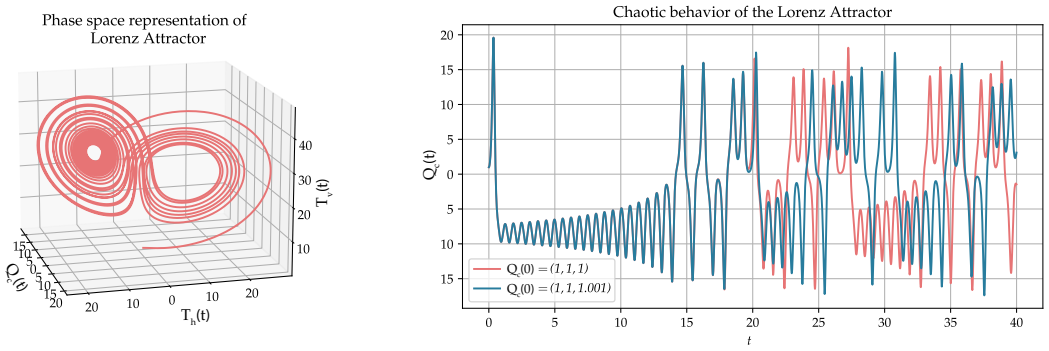
Solving equation (2-1) forward in time from the initial condition $\boldsymbol{s}(t{=}0)$ provides the temporal behavior of the system, also called **trajectory** $\boldsymbol{s}(t), t \geqslant 0$. Of course, the more complex the system, the more challenging it is to solve the ODE.

In best-case scenarios, the dynamics is linear, i.e. $\dot{\boldsymbol{s}} = \mathrm{A}\boldsymbol{s}$ where A is a constant matrix. Then the general solution of the ODE is

$$\boldsymbol{s}(t) = \boldsymbol{s}(t{=}0)e^{\mathrm{A}t}. \tag{2-2}$$

This solution exhibits different behaviors according to the eigenvalues of A. If all eigenvalues

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 2.1: **Lorenz Attractor** – (a) Behavior of a simulated trajectory using the Lorenz system for climate (σ=10, ρ=28, β=8/3) in the phase plane. The trajectory exhibits a "strange attractor" behavior, resembling the shape of a butterfly. It oscillates between two equilibrium points, creating an intricate pattern. (b) Evolution of the Q_c variable from two initial conditions very close to each other. As time increases, the trajectories start diverging, showcasing the chaotic behavior.

have negative real parts, the exponential converges to a constant (zero if real parts are strictly negative) as time increases, and the system is said to be *stable*. The smaller the eigenvalues, the faster the system converges. Conversely, if at least one eigenvalue has a positive real part, the exponential diverges towards infinity, and the system is *unstable*.

The linear form is convenient for analysis, but very rare in nature. Most systems require non-linear dynamics to truly represent the real world, such as saturation, hysteresis, or dead zone. This makes studying stability much more challenging. Consequently, many industrial applications linearize the dynamics near a setpoint $\boldsymbol{s}^*$ by defining the matrix of the dynamics as the jacobian of f at this point. However, the linearized dynamics is, by construction, only accurate in a small region near $\boldsymbol{s}^*$.

The Lorenz dynamics (Lorenz, 1963) is a good example of a seemingly simple system yet very challenging to work with. It was introduced by Edward Lorenz as a simplified model of the climate and involves three state variables

$$\begin{aligned} \dot{Q}_c(t) &= \sigma\big(T_h(t) - Q_c(t)\big), \\ \dot{T}_h(t) &= \rho Q_c(t) - T_h(t) - Q_c(t)T_v(t), \\ \dot{T}_v(t) &= Q_c(t)T_h(t) - \beta T_v(t), \end{aligned} \tag{2-3}$$

with $\sigma, \rho, \beta \in \mathbb{R}$ are positive parameters and $\boldsymbol{s} = \begin{pmatrix} Q_c & T_h & T_v \end{pmatrix}^\top$ is the state variable (Q_c: rate of convection, T_h: horizontal temperature and T_v: vertical temperature). This system is famous for its chaotic nature (see figure 2.1), characterized by its sensibility to initial conditions (the so-called *butterfly effect*). Interestingly, the trajectories of this system exhibit a *strange attractor* resembling a butterfly.

A physical system may also accept an external signal called the *control input* $\boldsymbol{u}(t)$. In

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

that case, the system is said to be *non-autonomous* and the command appears as input in the dynamics $f(\boldsymbol{s}, \boldsymbol{u})$. This signal is used to steer the system toward a wanted behavior, such as stabilization, reference tracking, regulation, etc. However, the inputs may be restricted on magnitude, or may not act on all state variables, making some states unreachable (referred as the *reachability* property). In that case, the system is said to be non *controllable*.

Finally, we usually do not have access to the complete state $\boldsymbol{s}(t)$ but rather to some indirect measurements $\boldsymbol{y}(t)$ obtained from sensors that may or may not be directly related to the state. This is modeled using a second equation added to the dynamical model

$$\begin{cases} \dot{\boldsymbol{s}}(t) &= f\big(\boldsymbol{s}(t), \boldsymbol{u}(t)\big), \\ \boldsymbol{y}(t) &= h\big(\boldsymbol{s}(t)\big). \end{cases} \tag{2-4}$$

The relation between the state and the observed variable is modeled by the *observation function* h. These observations can vary from simple quantities such as position/velocity measurements, voltages, and temperatures, to more intricate data. For instance, images from a camera often hold a lot of may information to reconstruct the state, but are difficult to exploit, as this information is entangled in the pixels. In that case, since the complete state is not directly measured, we leverage an *observer* algorithm to estimate the state from the observations. The Kalman filter is a popular example of observer.

The Luenberger observer (Luenberger, 1964) operates on linear systems $\dot{\boldsymbol{s}} = \mathrm{A}\boldsymbol{s} + \mathrm{B}\boldsymbol{u}$ with linear observation $\boldsymbol{y} = \mathrm{C}\boldsymbol{s}$. It copies the dynamics of the true system and adds a correction term related to the observation

$$\begin{cases} \dot{\hat{\boldsymbol{s}}} &= \mathrm{A}\hat{\boldsymbol{s}} + \mathrm{B}\boldsymbol{u} + \mathrm{L}(\boldsymbol{y} - \hat{\boldsymbol{y}}), \\ \hat{\boldsymbol{y}} &= C\hat{\boldsymbol{s}}. \end{cases} \tag{2-5}$$

The constant matrix L is chosen so that the error $\boldsymbol{e} = \boldsymbol{s} - \hat{\boldsymbol{s}}$ converges to zero, which can be achieved by setting L to stabilize the dynamics of $\dot{\boldsymbol{e}} = (\mathrm{A} - \mathrm{LC})\boldsymbol{e}$, i.e. $\mathrm{A} - \mathrm{LC}$ Hurwitz.

For a comprehensive review on observer design, see Bernard et al. (2022). However, the physical state may not be always retrievable from an observation. It may require a temporal window of measurements (estimating velocities from images), or be simply impossible, (estimating the torque of a motor from the voltage). In that case, we say that the system is not *observable*.

Ordinary differential equations are suitable for modeling physical quantities that depend on a single variable, typically time. However, they have limitations when it comes to capturing the behavior of fields or distributed systems that involve multiple dimensions. To address this, we turn to Partial Differential Equations (PDEs). In general, a PDE is formulated as

$$\dot{\boldsymbol{s}}(\boldsymbol{x}, t) = f(\boldsymbol{s}, \boldsymbol{u}, t, \boldsymbol{x}, \nabla \boldsymbol{s}, ..., \nabla^{(n)} \boldsymbol{s}). \tag{2-6}$$

Here, the equation involves a vector field $\boldsymbol{s}(\boldsymbol{x}, t)$ depending on a time variable t and a spatial variable $\boldsymbol{x}$. The dynamic of the system is not only dependent on the state $\boldsymbol{s}$ but also on its

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

partial derivatives with respect to $\boldsymbol{x}$, denoted by the spatial gradient operator $\nabla \boldsymbol{s}|_{ij} = \frac{\partial s_i}{\partial x_j}$ where s_i is the ith component of $\boldsymbol{s}$ and x_j is the jth spatial coordinate. Note that one can cast a PDE problem into an ODE by introducing an extended state $\underline{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s}(\boldsymbol{x}_1)^\top & \boldsymbol{s}(\boldsymbol{x}_2)^\top & \dots \end{bmatrix}^\top$ built from a set of discrete evaluation points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$.

2.1.2 *Contribution of data-driven approaches*

The state-space representation (equation (2-4)) is a very common and general framework and many algorithms for simulation and control are described within this setup. In this chapter, we focus on data-driven approaches for building and identifying parametric models of the dynamics. Specialized models (e.g. for simulation) will be discussed in the next chapter.

The conventional way of building a dynamical model relies on *ab initio* principles. In that case, interactions within the system are modeled from the fundamental laws of physics and involve parameters, such as diffusion coefficients, lengths, constants for motor power, etc. This identification step typically uses a dataset of N measured trajectories with the corresponding inputs $\mathcal{D} = \{(\boldsymbol{y}_n[k], \boldsymbol{u}_n[k] \mid n \in [\![1, N]\!]\}$ sampled at K discrete time instants $t_1, t_2, \cdots, t_K$. For an *ab initio* model, identification consists in solving an optimization problem, minimizing the distance between the ground truth observations and the predictions from the model.

During the early ages of machine learning for system identification, trajectories were approached as discrete time series. The simplest data-driven methods is the *Auto-Regressive model with eXternal inputs* (ARX) and takes the form of a linear equation with two matrix parameters (A, B):

$$\boldsymbol{y}[k] = \mathrm{A} \begin{bmatrix} \boldsymbol{y}[k-1]^\top & \boldsymbol{y}[k-2]^\top & \cdots & \boldsymbol{y}[k-\ell]^\top \end{bmatrix}^\top + \mathrm{B} \begin{bmatrix} \boldsymbol{u}[k]^\top & \boldsymbol{u}[k-1]^\top & \cdots & \boldsymbol{u}[k-\ell+1]^\top \end{bmatrix}^\top, \tag{2-7}$$

with $\boldsymbol{y}[k] = \boldsymbol{y}(t_k)$. ARX depends on a hyper-parameter ℓ corresponding to the temporal observation horizon. This model is very simple, but also very versatile and easy to identify by solving a least mean square problem. Note that ARX only involves observations and commands, hence the notion of state is not explicitly used. The temporal evolution is assumed to be causal, but non-markovian. The linear equation might be replaced by a non-linear parametric function for better performances (so-called Non-linear ARX models), yet at the cost of a more complicated identification step. For instance, Multi-Layer Perceptrons (MLPs) trained with gradient descent are a possibility. Finding the appropriate horizon can be related to *finite-time observability* properties, which corresponds to the smallest number of past observations required to reconstruct the state.

The notion of state is introduced with Recurrent Neural Networks (RNNs) (Rumelhart et al., 1985), where the state variable usually corresponds to as a hidden memory vector $\boldsymbol{z}$. The discrete dynamical equation leverages two weight matrices $\mathrm{W}_z, \mathrm{W}_v$ and one bias vector $\boldsymbol{b}$

$$\boldsymbol{z}[k+1] = \tanh\left(\mathrm{W}_z \boldsymbol{z}[k] + \mathrm{W}_v \boldsymbol{v}[k] + \boldsymbol{b}\right) \tag{2-8}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Historically, this model was designed for sequential aggregation of time series into a single vector. The hidden memory is initialized to an arbitrary value (typically zero) and is updated step-by-step using the input sequence $\{v[0], v[1], ...\}$. RNNs can be stacked into layers by connecting the hidden memory to the input of the next unit. Yet, it is known for suffering from vanishing gradients for long sequences, hence recent applications leverage more advanced designs such as Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014).

The RNN shares many similarities with the state-space representation of dynamical systems. Indeed, equation (2-8) can be seen as:

- **A non-autonomous dynamical system** in discrete time, with $z[k]$ being the state variable and $v[k]$ being the input control. In that case, the RNN can be used auto-regressively to forecast a trajectory of the memory vector. One might retrieve the observation with another projection network, such as an MLP.
- **The observer of an autonomous dynamical system** where the observation is the input of the RNN $v[k] = y[k]$ and the state estimate is $z[k]$. In that case, the RNN resembles the Luenburger observer (equation (2-5)), if we omit the hyperbolic tangent. The term $W_v v[k]$ can be seen as a correction term for the dynamics $z[k+1] = \tanh\left(W_z z[k] + b\right)$.

The relation between dynamical systems and recurrent neural networks has been studied since the early stages of deep learning (Funahashi and Nakamura, 1993; Draye et al., 1996) but also more recently (Chang et al., 2019a) for identification purposes (Wang and Chen, 2006; Schmidt et al., 2020) of potentially partially observed systems (Bhat and Munch, 2022) or control (Fang and Chow, 2005; Pan and Wang, 2011; Chow and Fang, 1998).

2.1.3 *Latent Dynamics*

Recurrent neural networks rely on the idea that there exists a latent space, generally of higher dimension than the observation, in which the dynamics becomes markovian. However, RNNs are arguably a rigid framework that might not be suited for all usage (Janny et al., 2021; Beintema et al., 2021). Thus a large body of work proposes more flexible structures to incorporate prior knowledge or enforce some properties. We can derive these models from a general framework which we refer to as *latent dynamics* (see figure 2.2)

$$s[k] = e_\theta\left(y[k], \cdots, y[k-\ell], u[k], ..., u[k-1], \cdots, u[k-\ell+1]\right), \tag{2-9}$$

$$\begin{aligned} s[k+1] &= f_\theta\left(s[k], u[k]\right), \\ y[k] &= h_\theta\left(s[k]\right). \end{aligned} \tag{2-10}$$

where e_θ, f_θ and h_θ are arbitrary functions with parameters θ. This framework can be interpreted in two different ways:

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

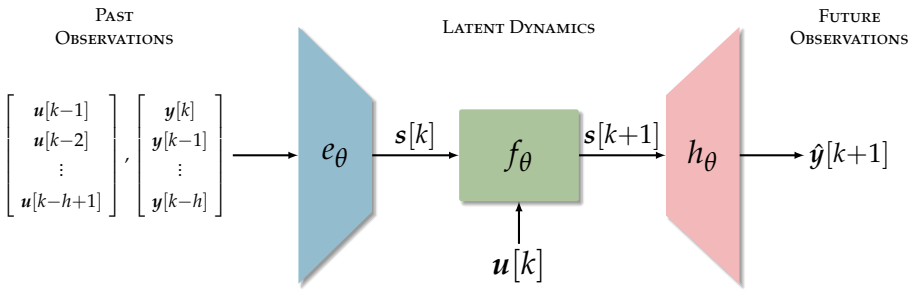


Figure 2.2: **Latent Dynamics** – is widely employed in system identification and encompasses a diverse set of neural network architectures. These models emulate the standard structure of a state-space model, where the system is observed solely through a measurement variable $\boldsymbol{y}$. To capture information from past observations, an encoder is employed to transform the input into a latent state vector $\boldsymbol{s}$, which exhibits flexible dynamics within the latent space. Subsequently, this latent state can be decoded back into the observation space using the observation function.

1. From a **deep learning** point of view, the couple of functions (e_θ, h_θ) can be seen as an auto-encoder (more precisely, a *trans-coder*) where $\boldsymbol{s}$ is the latent representation. The internal function f_θ is an auto-regressive model updating the latent state representation.

2. From a **control theory** point of view, equation (2-9) is a state observer using a finite horizon of previous observations to retrieve the state representation in latent space. The rest of the framework is directly a state-space model with its observation function.

Note that for a RNN in observer mode (inputs are the observation), the equation (2-9) involving the state observer is not used since the latent memory is initialized at zero. The dynamics of the RNN must converge to the correct state using solely the window of observation. The success of this setup is tied to the contraction property, which will be discussed in Chapter 5.

Deep state-space models Modeling a dynamics in a learned latent space is a very common practice in system identification. This approach has become ubiquitous in large-scale learning and the control community has built on this setup, leading to numerous variations of the same framework (Masti and Bemporad, 2018; Beintema et al., 2021; Rangapuram et al., 2018; Beintema et al., 2021; Gedon et al., 2021; Klushyn et al., 2021; Zhao et al., 2022).

Most general implementations model the encoder either with MLPs (for 1D observations), or with Convolutional Neural Networks (CNNs) (for 2D). One might also leverage conventional state observers (Forgione et al., 2022), such as a Kalman filter (Ruchti et al., 1993; Revach et al., 2022). The dynamics and observation function f_θ, h_θ (equation (2-10)) can also be modeled with MLPs, or any relevant structure for enforcing some properties or incorporate prior knowledge. We will present different examples in section 2.3.

The set of parameter θ can be identified from the dataset $\mathcal{D}$ of discrete trajectories and inputs by optimizing different losses corresponding to the desired behavior of the system. For simplicity, let $\boldsymbol{x}[i|j] = \begin{bmatrix} \boldsymbol{x}[i]^\top & \cdots & \boldsymbol{x}[j]^\top \end{bmatrix}^\top$ for all $i < j$ and $|\cdot|$ an adequate norm. The most

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

frequent training objectives are

- **The trans-coding loss** focuses on the behavior of e_θ and h_θ. This term encourages the model to maintain consistency in the latent space:

$$\mathcal{L}_{\text{transcode}} = \sum_{\boldsymbol{y},\boldsymbol{u}\in\mathcal{D}_d} \sum_{k=0}^{K} \left| \boldsymbol{y}[k] - h_\theta\Big(\underbrace{e_\theta\Big(\boldsymbol{y}[k|k-h], \boldsymbol{u}[k-1|k-h+1]\Big)}_{s[k]} \Big) \right|. \tag{2-11}$$

- **The observation forecasting loss** supervises the model end-to-end by computing the error between the next observation in time and the prediction from the model. The loss below is limited to a single prediction step but might be extended to longer trajectories, as this tends to improve the robustness of the model against error accumulation.

$$\mathcal{L}_{\text{forecasts}} = \sum_{\boldsymbol{y},\boldsymbol{u}\in\mathcal{D}_d} \sum_{k=0}^{K} \left| \boldsymbol{y}[k+1] - h_\theta \circ \overbrace{f_\theta\Big(\underbrace{e_\theta\big(\boldsymbol{y}[k|k-h], \boldsymbol{u}[k-1|k-h+1]\big)}_{s[k]}, \boldsymbol{u}[k] \Big)}^{s[k+1]} \right|, \tag{2-12}$$

  where $h_\theta \circ f_\theta(\cdot) = h_\theta\big(f_\theta(\cdot)\big)$.

- **The dynamics loss** ensures that the forecasted states in the latent space correspond to the state estimates from the encoder. The functions e_θ and f_θ are highly related, in the sense that the encoder corresponds to a state observer for the learned dynamics. This loss adds supervision over this relationship.

$$\mathcal{L}_{\text{dynamics}} = \sum_{\boldsymbol{y},\boldsymbol{u}\in\mathcal{D}_d} \sum_{k=0}^{K} \Bigg| \overbrace{e_\theta\Big(\boldsymbol{y}[k+1|k-h+1], \boldsymbol{u}[k|k-h]\Big)}^{s[k+1]} - f_\theta\Big(\underbrace{e_\theta\big(\boldsymbol{y}[k|k-h], \boldsymbol{u}[k-1|k-h+1]\big)}_{s[k]}, \boldsymbol{u}[k] \Big) \Bigg|. \tag{2-13}$$

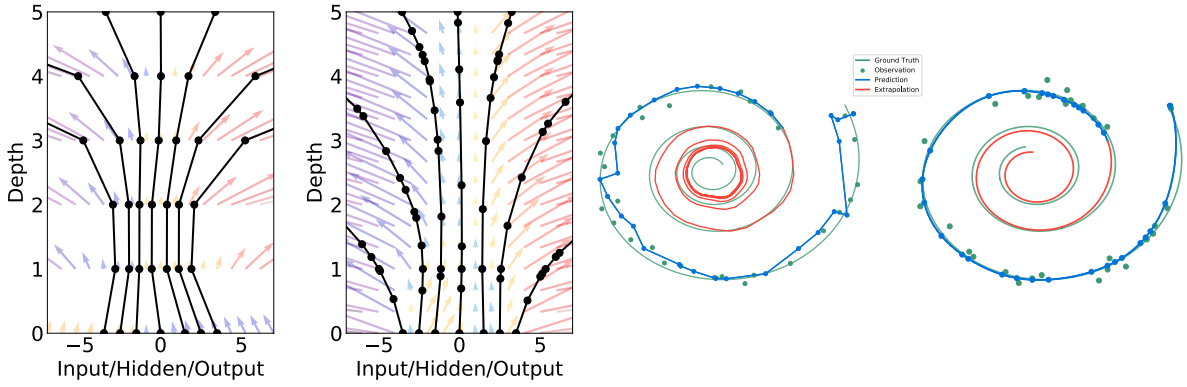
For an in-depth presentation, see for instance Masti and Bemporad (2018); Beintema et al. (2023). These losses can be combined with different weights and minimized using gradient descent. Other terms may appear to regularize different aspects of the dynamics. The latent dynamics framework is flexible and can be tailored to various problems. For instance, dynamics modeling in video analysis can leverage CNN-based functions f_θ (Beintema et al., 2021), while graph neural networks can be employed for tackling particle physics (Sanchez-Gonzalez et al., 2020). The adaptability of this approach enables its application to diverse domains and problem settings.

2.2 One Training Method to Rule Them All

2.2.1 *Solvers for ODE-based problems*

So far we have described data-driven methods for identifying discrete-time systems. In that case, the practical implementation of the dynamics and the training criterion are easier and

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b) (c)

Figure 2.3: **Neural ODE** – (a) Residual neural networks can be seen as the flow of a time-continuous ODE evaluated at fixed points in time (black dots), (b) Neural ODE, using a different numerical solver, enables querying the dynamics at intermediate points, leading to improved flow estimation. (c) Forecasted trajectory of a damped pendulum using RNN and Neural ODE. Both models are trained from non-uniformly sampled noisy state observations. Neural ODE produces smoother results and better time extrapolation (figures from Chen et al. (2018)).

can be derived readily from the equations. However, there are many advantages to targeting continuous-time dynamics. At first glance, nature appears to be continuous (omitting quantum mechanics) and most of the laws of physics are described in continuous time. Moreover, a larger set of the bibliography in control theory and physics is dedicated to continuous time systems rather than discrete time. Hence, more algorithms and methods are available.

However, training a neural network to model a continuous-time state-space representation is not straightforward. It raises two important questions: (1) how can we implement a continuous model numerically? and (2) how can we train it from a discrete set of observations $\mathcal{D}$? A possibility is to draw inspiration from discretization schemes for ODEs. The most popular method is the first-order Runge-Kutta (Euler) method (Euler, 1824)

$$\boldsymbol{s}(t+\delta) = \boldsymbol{s}(t) + \delta \frac{\partial \boldsymbol{s}}{\partial t} + O(\delta^2) \approx \boldsymbol{s}(t) + \delta f_\theta(\boldsymbol{s}, \boldsymbol{u}, t), \tag{2-14}$$

where δ is a small time step. Numerically solving ODEs is not trivial: non-linearities, stiffness, and chaotic behaviors can lead to poor numerical stability and divergence of the simulation. Higher-order methods (Runge, 1895; Kutta, 1901) offer improved accuracy and stability compared to the simple Euler method, still suffer from numerical issues when dealing with stiff systems, requiring adaptive step sizes and implicit solvers (Hairer et al., 1993).

Interestingly, the formula (2-14) resembles residual layers in deep learning, as seen in models developed for computer vision (ResNet (He et al., 2016)), where the input z_ℓ of a layer ℓ is connected to its output $g_\theta(z_\ell)$, such that

$$z_{\ell+1} = z_\ell + g_\theta(z_\ell). \tag{2-15}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Neural Ordinary Differential Equations In Chen et al. (2018), the authors acknowledge the similarity between a residual layer and the explicit Euler discretization scheme. They propose using a more powerful algorithm, introducing the concept of *neural ordinary differential equations* (see Figure 2.3). The purpose of this model goes beyond dynamical systems and can be used for various tasks, including classification. The model assumes that the latent state follows an ODE. The output of a Neural-ODE, given initial and final instants t_0 and t_1, is expressed as

$$z(t_1) = \text{ODESolve}(z_0, g_\theta, t_0, t_1), \tag{2-16}$$

where ODESolve refers to any numerical solver, such as a higher-order Runge-Kutta (RK) method (pioneered in Wang and Lin (1998)), and g_θ is a neural network. However, this method can be computationally demanding, particularly for high-resolution, as it requires back-propagating gradients through the ODE solver. To address this, the authors employ the **adjoint sensitivity method** (Pontryagin, 1987), to reduce computational complexity. The adjoint method efficiently computes gradients by combining the original ODE with an adjoint ODE, which can then be solved using the same solver.

Neural-ODE has paved the way to various follow-up work, leveraging this insight from two different angles:

- **Dynamical system perspectives of deep learning**: Neural-ODE casts residual neural networks into a continuous process, each layer being an iteration of an ODE solver. By reducing the time step, we can artificially increase the number of layers, yielding neural networks with theoretically *infinite number of layers*. It opens the door for using tools from control theory to study neural networks (Rodriguez et al., 2022; Chang et al., 2019a; Haber and Ruthotto, 2017).

- **Deep learning applied to dynamical systems**: Neural-ODEs can also be embedded into the latent dynamics setup. It allows modeling the continuous dynamics f with a neural network in a latent space, which is *discovered* during training. The dynamics is trained end-to-end from a discrete set of measures $\mathcal{D}$ while keeping the advantages of continuous formulation and explicit solvers. It also facilitates the incorporation of prior knowledge from physics (see next section). For instance, Neural-ODEs can be enhanced to handle second-order dynamics (Dupont et al., 2019; Norcliffe et al., 2020), stiff dynamics (e.g. bouncing objects (Jia and Benson, 2019; Chen et al., 2020; Poli et al., 2021)) or graph-related problems (Chamberlain et al., 2021; Poli et al., 2019).

In what follows, we are interested in the second interpretation. We can leverage Neural-ODE as a drop-in replacement for the discrete dynamical system in the latent dynamics (equation (2-9) and (2-10)) such that the dynamics f_θ is explicitly modeled in continuous time. In practice, we use an ODE solver to compute trajectories at discrete time steps.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

2.2.2 *Solvers for PDE-based problems*

Handling continuous dynamics for PDEs is an even greater challenge compared to ODEs, as it involves discretizing multiple variables. To derive a training technique, we can once again look at conventional discretization schemes and embed our dynamical model into it.

The simplest scheme for solving PDEs uses a uniform grid to divide the spatial domain into a tiled arrangement of points. The finer the spacing between points, the more accurate the model becomes. Yet, this method, known as the *finite difference method* and inspired by the Euler method for ODEs, is limited to simple geometries with low curvature. Indeed, maintaining uniformity across the entire domain constrains resolution, with direct impacts on the computation time: if a specific region requires a finer mesh, it impacts the entire simulation, and substantially increases the number of points to process.

To tackle these challenges, non-uniform grids offer greater flexibility. Two popular methods are the Finite Element Methods (FEMs) and the Finite Volume Methods (FVMs) which allow for local mesh refinement to improve accuracy near critical areas while reducing computational cost elsewhere.

- FEM discretizes the domain into a mesh of elements, such as triangles or quadrilaterals in 2D and tetrahedra or hexahedra in 3D. The trajectory is approximated by basis functions defined on these elements, minimizing the residual of the PDE.
- FVM divides the domain into control volumes, often based on the dual mesh of the FEM mesh. The method approximates the PDE by integrating the equation over each control volume and leveraging the divergence theorem to establish relationships between solution values at control volume faces. FVM finds extensive applications in fluid dynamics and heat transfer problems.

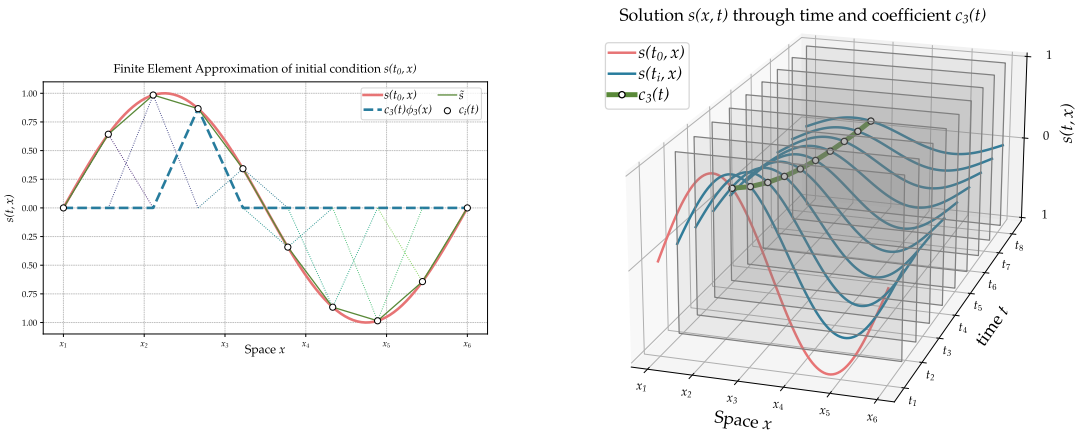
These techniques are significantly more advanced than the one used for ODEs. To this day, most of system identification techniques based on deep learning for PDEs still rely on discrete setup. Yet, recent work attempts to leverage these algorithms to learn continuous dynamics, following the insights from Neural-ODE.

Finite Element Networks Let us consider the FEM and consider a discretization of the spatial domain $\mathcal{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ...)$. The method proceeds by decomposing the solution on a basis of local functions ϕ_j such that $\phi_j(x_i)$=1 if i=j and zero otherwise. The discrete solution on $\mathcal{X}$ depends on coefficients $c_j(t)$ such that

$$\boldsymbol{s}(\boldsymbol{x}_i, t) = \sum_j c_j(t)\phi_j(\boldsymbol{x}_i). \tag{2-17}$$

The **Galerkin method** (Galerkin, 1915) seeks a solution making the residual $R(\boldsymbol{s}) = \dot{\boldsymbol{s}} - f(\boldsymbol{s}, t, \boldsymbol{x}, \nabla \boldsymbol{s}, \cdots)$ orthogonal to the space spanned by the basis functions, i.e. $\langle R(\boldsymbol{s}), \phi_i \rangle = 0$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 2.4: **Finite Elements Networks** – (a) the solution $\boldsymbol{s}(t, \boldsymbol{x})$ of a PDE can be decomposed at discrete points in space $x_1, x_2, ...$ on basis functions ϕ_i linear by part, weighted by some coefficients c_i. (b) The Galerkin method consists of solving an ODE in the coefficients $c_i(t)$ (figure adapted from Lienen and Günnemann (2022)).

for all i. This gives a condition on the coefficients $c_i(t)$ which takes the form of an ODE and thus can be addressed using Neural-ODE (See figure 2.4 for an overview). Finite Element Networks (Lienen and Günnemann, 2022) draw inspiration from these concepts and locally model the dynamics f using a neural network. This approach harnesses the inductive biases from FEM, enhancing the performance of learning-based physics simulators. In a similar vein, a related derivation based on the FVM can be found in Karlbauer et al. (2022).

2.3 The Fellowship of the Dynamical Systems

By associating the latent dynamics model with a Neural-ODE-like training method, we are now ready to learn a dynamical model from the observed trajectories from the dataset $\mathcal{D}$. In a sense, this setup already embeds prior knowledge, by modeling the dynamics in a higher dimensional space as a differential equation. Yet, it may be beneficial to incorporate additional information on the system of interest. In this section, we discuss different types of structures that can be used to model the dynamics f_θ, taking into account the type of problem we are addressing and the requirements we would like to satisfy.

2.3.1 *Structured linear dynamical systems*

Linear models are a very convenient structure since studying linear systems is considerably simpler. Their analytical solution is known, stability can be assessed by examining the eigenvalues of the matrix operator, and properties such as observability and controllability can be determined through relatively straightforward linear algebra. Moreover, many controllers and observers can be readily implemented, and simulation can be achieved efficiently. Such models can be identified with latent dynamics involving three matrices A, B and C to model

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

the dynamics, without constraints on the encoder

$$\begin{cases} \dot{\boldsymbol{s}} &= f_\theta(\boldsymbol{s},\boldsymbol{u}) &= A\boldsymbol{s}+B\boldsymbol{u}, \\ \boldsymbol{y} &= h_\theta(\boldsymbol{s}) &= C\boldsymbol{s}. \end{cases} \tag{2-18}$$

Note that we are using a continuous time setup here, which can be embedded into a Neural-ODE, or replaced by discrete-time dynamics. This setup has been explored for instance in Masti and Bemporad (2018), which leverages a linear time-variant model, where the parameters A, B and C depend on time and are computed via a dedicated MLP.

However, one might be concerned by the existence of a suitable latent space where the dynamic of the studied system can indeed be linear. Koopman theory provides insights into this matter.

Consider an autonomous discrete dynamical system (i.e. without external inputs) $\boldsymbol{s}[k+1] = f(\boldsymbol{s}[k])$ where $\boldsymbol{s} \in \mathbb{R}^n$. In an infinite-dimensional Hilbert space, there exist *observable* functions $g : \mathbb{R}^n \to \mathbb{R}$ for which the **Koopman operator** $\mathcal{K}$ advances the function in time, i.e.

$$\mathcal{K}g = g \circ f \Leftrightarrow \mathcal{K}g(\boldsymbol{s}[k]) = g(\boldsymbol{s}[k+1]). \tag{2-19}$$

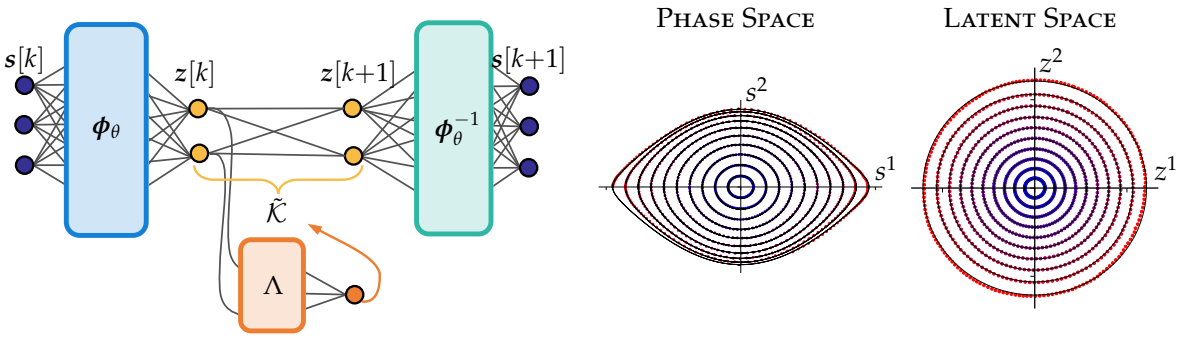
The Koopman operator reveals the existence of a universal linear embedding space for non-linear systems. However, linearity comes at the cost of infinite dimensionality. Theoretical studies on the Koopman operator can be found in Mezić and Banaszuk (2004); Mezić (2005); Budišić et al. (2012). This framework paves the way for exploring linear dynamics in learned latent spaces discovered from data. Indeed, the linearity of the Koopman operator yields the existence of eigenfunctions, which corresponds to an infinite set of measurement functions evolving linearly in time such that

$$\varphi(\boldsymbol{s}[k+1]) = \mathcal{K}\varphi(\boldsymbol{s}[k]) = \lambda\varphi(\boldsymbol{s}[k]). \tag{2-20}$$

Hence, Koopman theory states the existence of a infinite-dimensional state-space $\boldsymbol{z}[k] = \varphi(\boldsymbol{s}[k])$ in which the dynamics is linear, i.e. $\boldsymbol{z}[k+1] = \mathcal{K}\boldsymbol{z}[k]$. Note that despite the change of notation, this is still aligned with the latent dynamics, the observation being the entire state with non-linear dynamics $\boldsymbol{s}$, and the latent state variable being $\boldsymbol{z}$ which exhibits a linear dynamics.

Deep Koopman Operator In Lusch et al. (2018), the Koopman operator is approximated using a finite-dimensional block-diagonal matrix $\tilde{\mathcal{K}}$. The input state $\boldsymbol{s}[k]$ is projected to a latent space with a MLP $\boldsymbol{z}[k] = \boldsymbol{\phi}_\theta(\boldsymbol{s}[k])$, where $\boldsymbol{\phi}_\theta$ models a vectorized eigenfunction (that is, a stack of multiple eigenfunctions φ). Within this latent space, the dynamics can be approximated by a linear function, following Koopman theory: $\boldsymbol{z}[k+1] = \tilde{\mathcal{K}}\boldsymbol{z}[k]$. Trajectories in the latent space are then decoded using a second MLP learning the inverse mapping ${\boldsymbol{\phi}_\theta}^{-1}$. Depending on the problem, the approximated Koopman operator can be modeled as a static matrix with learnable diagonal terms or parametrized by an auxiliary network

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 2.5: **Deep Koopman operator** – (a) The Koopman operator is a linear operator in infinite dimension. The model proposes to learn the eigenfunctions of this operator to project the states into a space where the dynamic becomes linear. (b) Example of a damped pendulum. On the left, the simulated trajectory in physics space (dots) remains close to the physical trajectory of the system (lines). On the right, trajectories in latent space where the dynamics is indeed linear. (figures from Lusch et al. (2018))

$[\lambda_{1,k}, \lambda_{2,k}, ...] = \Lambda_\theta(\boldsymbol{z}[k])$ to handle continuous spectra:

$$\tilde{\mathcal{K}} = \text{diag}\big(B(\lambda_{1,k}), B(\lambda_{2,k}, ...)\big), \quad B(\lambda{=}\mu + i\omega) = \exp(\mu) \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \tag{2-21}$$

The model (shown in figure 2.5) is trained to optimize three classic objectives for latent dynamics over a prediction horizon $m \in \mathbb{N}^+$:

$$\text{Auto-encoder:} \quad \min \left| \boldsymbol{s}[k] - \boldsymbol{\phi}_\theta{}^{-1}\Big(\boldsymbol{\phi}_\theta\big(\boldsymbol{s}[k]\big)\Big) \right| \tag{2-22}$$

$$\text{Linear dynamics:} \quad \min \left| \boldsymbol{\phi}_\theta\big(\boldsymbol{s}[k{+}m]\big) - \tilde{\mathcal{K}}^m \boldsymbol{\phi}_\theta\big(\boldsymbol{s}[k]\big) \right| \tag{2-23}$$

$$\text{Prediction:} \quad \min \left| \boldsymbol{s}[k{+}m] - \boldsymbol{\phi}_\theta{}^{-1}\Big(\tilde{\mathcal{K}}^m \boldsymbol{\phi}_\theta\big(\boldsymbol{s}[k]\big)\Big) \right| \tag{2-24}$$

The Koopman operator has garnered significant attention in recent research (Han et al., 2020; Yeung et al., 2019; Li and Jiang, 2021; Mardt et al., 2020). We believe its popularity stems from two key factors. First, it offers a robust theoretical framework for studying dynamics in a linear representation, a domain well-understood by researchers. Secondly, while analytically calculating the eigenfunctions of the operator is exceptionally challenging (Mezić, 2005), training-based methods have emerged as a viable alternative. A concurrent approach will however be presented in detail in Chapter 5: the Kazantzis-Kravaris-Luenberger observer.

2.3.2 *Latent dynamics for mechanics*

One can derive a latent dynamics tailored to mechanical systems. The traditional approach, known as Newtonian mechanics, employs vectors to describe position, velocity, and forces. However, this vector representation often disregards mechanical constraints. For instance, when studying a pendulum, it suffices to analyze the evolution of the angle rather than the 3D movements of the center of mass.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

In contrast, analytical mechanics characterizes motion using scalar quantities that inherently account for the degrees of freedom. Derived from the principle of minimal action, various theories exist within this framework. *Lagrangian* mechanics, for instance, incorporates constraints directly in a set of generalized coordinates $\boldsymbol{q} = [q_1(t), q_2(t), ...]$ governed by the Euler-Lagrange equation

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = T(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) - V(\boldsymbol{q}, \dot{\boldsymbol{q}}, t), \qquad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) = \frac{\partial L}{\partial \boldsymbol{q}}. \tag{2-25}$$

Here, $T(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ represents the total kinetic energy and $V(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ denotes the total potential energy. Similarly, by introducing the generalized momentum $\boldsymbol{p} = \frac{\partial L}{\partial \dot{\boldsymbol{q}}}$, we can derive the equations of *Hamiltonian* mechanics

$$H(\boldsymbol{q}, \boldsymbol{p}, t) = \boldsymbol{p} \cdot \dot{\boldsymbol{q}} - L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t), \qquad \dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}}, \qquad \dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}}, \tag{2-26}$$

where H is the Hamiltonian of the system.

Lagrangian/Hamiltonian Neural Network Learning a Hamiltonian dynamical system from data can be achieved by employing a neural network to model $H_\theta(\boldsymbol{q}, \boldsymbol{p}, t)$. For example, Greydanus et al. (2019) proposes a training approach using the following cost function

$$\mathcal{L}_{\text{HNN}} = \left| \frac{\partial H_\theta}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{q}}{\partial t} \right| + \left| \frac{\partial H_\theta}{\partial \boldsymbol{q}} + \frac{\partial \boldsymbol{p}}{\partial t} \right|, \tag{2-27}$$

for some norm $|\cdot|$. Hamiltonian neural networks enforce conservation laws and demonstrate improved performance in simulating dynamical systems that typically suffer from energy loss when using Newtonian models. However, this approach requires access to ground truth generalized coordinates, which may pose challenges. Extensions have been developed to handle arbitrary coordinates (Choudhary et al., 2021), image-based observations (Toth et al., 2019), and non-autonomous systems (Zhong et al., 2019).

In contrast, Lagrangian dynamics can accommodate arbitrary coordinates by modeling the Lagrangian function $L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$. Notably, Lutter et al. (2019); Cranmer et al. (2020b) reformulate equation (2-25) as follows

$$\ddot{\boldsymbol{q}} = \left(\nabla_{\dot{\boldsymbol{q}}} \nabla_{\boldsymbol{q}}^T L_\theta\right)^{-1} \left[\nabla_{\boldsymbol{q}} L_\theta - \left(\nabla_{\boldsymbol{q}} \nabla_{\dot{\boldsymbol{q}}} L_\theta\right) \dot{\boldsymbol{q}}\right]. \tag{2-28}$$

Here, L_θ is a neural network approximating the Lagrangian function. Although this equation may appear complex, it can be numerically computed using automatic differentiation, enabling simulation and back-propagation of the gradient to minimize the prediction error. These formulations have been applied to graph Neural-ODEs in Bishnoi et al. (2023), and to images in Zhong and Leonard (2020).

Despite the advantages offered by these methods, their current applications remain limited to simpler systems. The main challenge lies in optimizing a cost function that depends

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

on the derivative of the neural network, necessitating multiple passes through the computation graph. While this is feasible, it may somewhat restrict the complexity of the underlying models.

2.3.3 *Stable dynamics*

Stability is a very useful property of dynamical systems and can be either an inductive bias (for systems known to be stable) or simply a relevant property for downstream tasks, such as simulation. The concept of stability varies in meaning between machine learning and control theory[1]. The former usually refers to the stability of the training algorithm, while the latter focuses on the behavior of the dynamical system. In the first section of this chapter, we introduced a criterion to verify the stability of linear systems, by checking the real part of the eigenvalues of the dynamics matrix (see equation (2-2)). In the other hand, the stability of non-linear systems is harder to verify and requires a Lyapunov analysis.

Definition 2.1 *An autonomous system* $\dot{\boldsymbol{s}} = f(\boldsymbol{s})$ *with an equilibrium point* $\boldsymbol{s}^* = f(\boldsymbol{s}^*)$ *is said to be stable if and only if*

$$\forall \delta > 0, \exists \epsilon > 0 \text{ such that } \left|\boldsymbol{s}_0 - \boldsymbol{s}^*\right| < \delta \Rightarrow \left|\boldsymbol{s}(t) - \boldsymbol{s}^*\right| < \epsilon, \ \forall t > 0. \tag{2-29}$$

To prove the Lyapunov stability of a dynamical system, one can show the existence of a real-valued *Lyapunov function* $V(\boldsymbol{s})$ such that (assuming $\boldsymbol{s}^* = 0$[2])

$$V(\boldsymbol{s}) = 0 \Leftrightarrow \boldsymbol{s} = 0 \qquad \forall \boldsymbol{s} \neq 0, \ V(\boldsymbol{s}) > 0 \qquad \dot{V}(\boldsymbol{s}) \leqslant 0. \tag{2-30}$$

The Lyapunov function is analogous to the potential energy of the system: it remains positive and reaches zero at the equilibrium point. The requirement for its derivative to be non-positive guarantees that the system gradually dissipates "energy" over time, ensuring stability. Thus, to assess system stability, it “suffices” to find a suitable Lyapunov function that satisfies (2-30). However, identifying such a function for nonlinear systems is a complex task, presenting significant challenges.

Stable deep dynamical models A promising challenge is thus to constrain a latent dynamics to learn a stable dynamical model. This can be achieved by learning a suitable Lyapunov function jointly with the dynamics. For instance, in Kolter and Manek (2019), the candidate Lyapunov function V is modeled using an input-convex neural network (ICNN) g_θ (Amos et al., 2017) to prevent local optima, and is defined as follow

$$V_\theta(\boldsymbol{s}) = \sigma\big(g_\theta(\boldsymbol{s}) - g_\theta(0)\big) + \epsilon|\boldsymbol{s}|^2, \tag{2-31}$$

where σ is a positive convex function with $\sigma(0) = 0$ and ϵ a small constant. Consequently,

[1] and is arguably on top of the list of misleading terms that should be carefully avoided during trans-disciplinary meetings

[2] in practice, any equilibrium point can be considered using a change of coordinates.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

the candidate Lyapunov function is convex and positive definite. The learned dynamics can then be projected onto a subspace where $\dot{V}_\theta(\boldsymbol{s}) \leqslant -\alpha V_\theta(\boldsymbol{s})$, α being a positive constant. This condition guarantees *global asymptotic stability*[a]. The dynamics f_θ is modeled with an MLP. The actual dynamics used during training is a projection of f_θ on the stable manifold defined by V_θ, i.e.

$$\tilde{f}_\theta(\boldsymbol{s}) = \begin{cases} f_\theta(\boldsymbol{s}) & \text{if } \dot{V}_\theta(\boldsymbol{s}) \leqslant -\alpha V_\theta(\boldsymbol{s}), \\ f_\theta(\boldsymbol{s}) - \frac{\nabla V_\theta(\boldsymbol{s})}{\|\nabla V_\theta(\boldsymbol{s})\|_2^2}\left[\nabla V_\theta(\boldsymbol{s})^T f_\theta(\boldsymbol{s}) + \alpha V_\theta(\boldsymbol{s})\right] & \text{otherwise.} \end{cases} \tag{2-32}$$

[a] to be precise, V_θ must also be radially unbounded

Note that this approach does not impose any additional properties on the Lyapunov function. Instead, the learned dynamics is directly projected onto a stable manifold, and the shape of V_θ is guided by the training objective. In another work, (Gaby et al., 2022) assume knowledge of the dynamics f_θ and seek an appropriate Lyapunov function by minimizing the following cost

$$\mathcal{L} = \sum_{\boldsymbol{s}} \max\left(\left[\nabla V_\theta(\boldsymbol{s}) \cdot f_\theta(\boldsymbol{s}) + \gamma|\boldsymbol{s}|\right]^2, 0\right). \tag{2-33}$$

Moreover, in Petridis and Petridis (2006), a neural candidate function is learned using genetic algorithms while satisfying certain constraints on the Hessian. Quadratic functions have also been employed to ensure positive definiteness (Abate et al., 2020; Chang et al., 2019b). However, stability is not always a desired property: for a drone (without taking into account collisions with the ground), the dynamical system is naturally unstable: without any inputs, the drone simply falls downwards indefinitely, and it is pointless to try to identify a stable model for it.

2.4 The Two Approaches: hybrid models

So far, we have presented structures based upon well-chosen frameworks, such as the lagrangian mechanics, linear systems, or stable dynamics, which embed prior knowledge on the *structure* of the dynamics. However, in many cases, we may have access to explicit knowledge of at least a part of the dynamical equation, obtained from first principles considerations and laws of physics.

Thus, it is tempting to incorporate the known part of the ODE in our learned model. To address this, *hybrid models* combine prior knowledge with data-driven approaches. Mathematically, we can decompose the true dynamics of a physical system f into a known component f_θ, derived from physics-based considerations, and an unknown residual term $\mathcal{R}$. By accessing measurements from the true system stored in a dataset $\mathcal{D}$, a hybrid model approximates the residual dynamics by modeling $\mathcal{R}$ with another model, such as a neural network g_ψ. The parameters of the known component θ and the residual ψ can be identified by minimizing the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

loss over $\mathcal{D}$

$$\arg_{\psi,\theta} \min \sum_{s \in \mathcal{D}} \mathcal{L}(s, \hat{s}) \quad \text{s.t} \quad \begin{cases} \dot{\hat{s}} & = f_\theta(\hat{s}) + g_\psi(\hat{s}) \\ \hat{s}(0) & = s(0) \end{cases}, \tag{2-34}$$

for some cost function $\mathcal{L}$ measuring the discrepancy between trajectories. Hybrid models offer several advantages:

- **Inductive bias** – Incorporating a handcrafted model, imparts a desired system behavior and provides an inductive bias for learning. This utilization of prior knowledge enhances generalization to novel scenarios beyond the training data, as helps capturing fundamental principles governing the system.
- **Interpretability** – Hybrid models offer interpretability, as the known component f_θ is based on well-understood physical principles. This enhances model diagnosis and facilitates understanding the contribution of the data-driven part. Moreover, hybrid models tend to exhibit robustness against noise and uncertainties in the training data.
- **Easier training** – Hybrid models benefit from data efficiency. The incorporation of a prior simplifies the training process, enabling the neural network to focus solely on learning the residual dynamics, which often reduces the need for larger datasets. This can lead to faster convergence, reduced risks of overfitting, and avoidance of local optima.

Early research in artificial neural networks introduced the concept of hybrid models, also known as gray-box models (Psichogios and Ungar, 1992; Rico-Martinez et al., 1994; Thompson and Kramer, 1994). These models offer great versatility, as they allow the integration of prior knowledge from various domains of physics, such as power systems (Nutkiewicz et al., 2018; Wang et al., 2019a) and fluid mechanics (Belbute-Peres et al., 2020; Young et al., 2017).

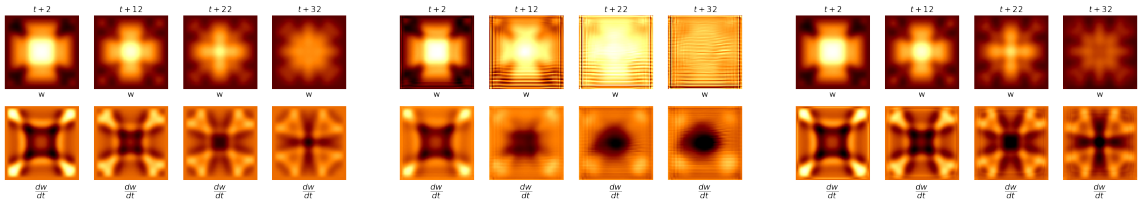
APHINITY Formally, let us consider the true dynamics $\dot{s} = f(s)$ where $s \in \Omega$ and f lies in some functional space $\mathcal{F}$. A hybrid model is the sum of a physics prior $f_\theta \in \mathcal{F}_p$ and a data-driven augmentation term $g_\psi \in \mathcal{F}_a$, where (θ, ψ) are parameters to be optimized such that $f = f_\theta + g_\psi$.

However, the decomposition is not unique, and without constraints on the parameters, the augmentation term can dominate the dynamics, overpowering the physics prior. Ideally, we desire the augmentation term g_ψ to solely capture the residual dynamics. For instance, when $f \in \mathcal{F}_p$, we expect the augmentation term to be identically zero. To address this, Yin et al. (2021b) seeks for a minimal norm decomposition of the form

$$\min_{f_\theta \in \mathcal{F}_p, g_\psi \in \mathcal{F}_a} |g_\psi| \quad \text{s.t} \quad \forall s \in \Omega, \dot{s} = f_\theta(s) + g_\psi(s). \tag{2-35}$$

The authors show the existence of a solution to equation (2-35) by assuming that $\mathcal{F}_p$ is

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) Ground Truth (b) Neural ODE (Chen et al., 2018) (c) APHINITY (Yin et al., 2021b)

Figure 2.6: **Hybrid dynamics** – Simulation from APHINITY with the hybrid dynamics of a damped wave equation. The physics prior consists of the wave equation without the damping factor. Each model is trained on trajectories of 25 frames, hence the last frame displayed here corresponds to the out-of-distribution temporal domain. The APHNITY model benefits from strong prior knowledge on the dynamics of the system, and performs better than a physics-agnostic Neural-ODE (figures from Yin et al. (2021b))

proximinal, which means that any function $x \in \mathcal{F}_p$ must have at least one closest neighbor. More precisely, for any subset $C \subset \mathcal{F}_p$, the set $P_C(x) = \{\boldsymbol{y} \in C || x - y| = d(x, C)\}$ is non-empty, where $d(x, C) = \arg\min_{c \in C} |x - c|$. Additionally, the solution is unique when $\mathcal{F}_p$ is a Chebyshev set, implying that $P_C(x)$ contains a single element for all $x \in \mathcal{F}_p$.

For practical implementations, neural networks are employed to model the augmentation term g_ψ. The dynamical model is trained using a dataset of measurements $\mathcal{D}$ and aims to minimize the objective function

$$\mathcal{L} = \sum_{s \in \mathcal{D}} \left| g_\psi(\boldsymbol{s}) \right| + \lambda \sum_{k=1}^{K} \left| \boldsymbol{s}[k] - \hat{\boldsymbol{s}}[k] \right|, \tag{2-36}$$

where λ is a weight that balances the importance of each term. The simulated trajectory $\hat{\boldsymbol{s}}$ can be obtained by solving the hybrid model numerically, starting from the known initial condition $\boldsymbol{s}(t{=}t_0)$ (e.g., using Neural-ODE). Figure 2.6 shows a simulation of a damped wave equation $\ddot{\boldsymbol{s}} = c^2 \Delta \boldsymbol{s} - k\boldsymbol{s}$, where the damping factor is removed from the equation to obtained the physics prior f_θ. APHINITY is compared to a physics-agnostic Neural-ODE model (i.e. without a handcrafted model in the dynamics). The model shows excellent performance on this simple problem.

Notably, the APHINITY framework has been extended and adapted in different ways. A variational auto-encoder variant was proposed in Takeishi and Kalousis (2021), while Mehta et al. (2021b) modified g_ψ by incorporating a latent memory of previous states, and Ajay et al. (2018) leverages stochastic neural networks to augment physics simulators.

It is worth highlighting that the incorporation of prior knowledge into learning models extends beyond residual learning from known analytical models. We have identified three general methods for achieving this integration:

- **Residual learning**, as mentioned earlier, involves combining an explicit physics model with a data-driven model to improve performance.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

To simulate the dynamics of a drone, a simple model based on kinematics and momentum conservation can be used, assuming proportionality between thrust and the squared rotation speed (Mahony et al., 2012). While this model suffices for control applications, it lacks accuracy for simulation tasks. Even with accurate identification and matching inputs, the simulation diverges from the real trajectory. Advanced modeling techniques, such as blade-element momentum (Prouty, 1995), offer better rotor modeling but still fall short in simulation accuracy. To address this, Bauersfeld et al. (2021) propose an approach combining APHINITY-like modeling with the blade-element momentum model. This hybrid model is trained using highly accurate drone flight data (captured with motion capture) and demonstrates exceptional performance in closed-loop flights, closely matching the real device trajectory.

- **Physics-guided design** of neural networks consists of adapting data-driven architectures to address specific physics problems. By leveraging prior knowledge about the system, the model structure can be tailored to incorporate relevant reasoning patterns. This can be achieved by incorporating additional terms in the model structure or by directly deducing the architecture from the task setup.

  Tracking surface temperature in oceans, denoted as $T(\boldsymbol{x}, t)$, requires considering fluid transport, thus involving advection and diffusion. If the physical equation governing the evolution of the temperature is particularly complex, it can be simplified by assuming the displacement field $w(\boldsymbol{x}, t)$ of the fluid to be known. The temperature then obeys the following PDE

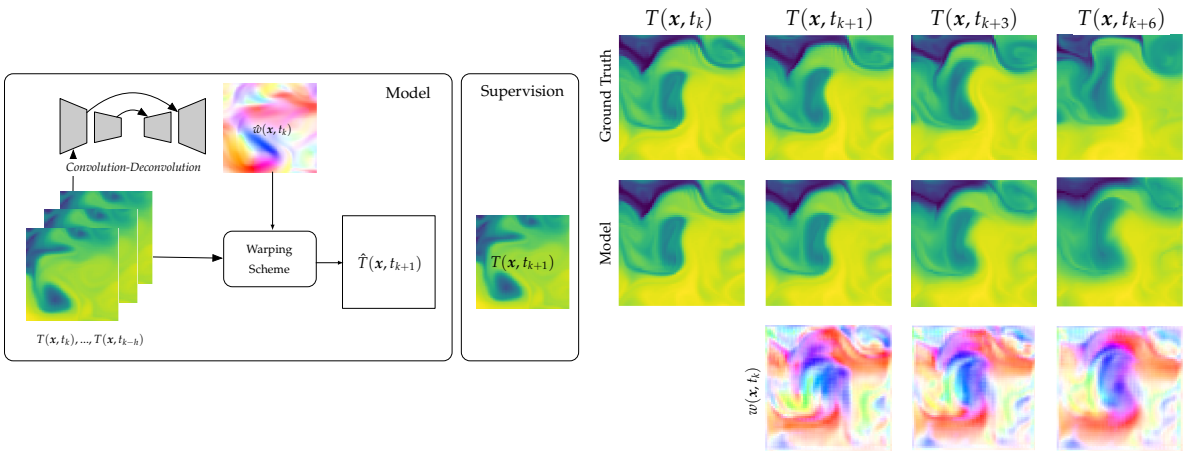
  $$\frac{\partial T}{\partial t}(\boldsymbol{x}, t) + (w(\boldsymbol{x}, t) \cdot \nabla)\, T(\boldsymbol{x}, t) = D\Delta T(\boldsymbol{x}, t)\,. \tag{2-37}$$

  This formulation bears similarities with optical flow in computer vision. In Bézenac et al. (2019), the equation is directly embedded in the model. They employ a CNN to forecast the displacement map $w(\boldsymbol{x}, t)$ from previous temperature maps, which is then used for updating the temperature using equation (2-37). The architecture and a sample of the results are illustrated in figure 2.7.

- **Differentiable physics simulators** offer a distinct approach compared to previous methods, as they do not directly aim to identify dynamics models but rather leverage differentiable physics simulators to incorporate priors into learning. This approach is widely used in robotics, particularly for object manipulation tasks.

  In situations where an agent needs to learn how to interact with its environment, the most frequent approach is to rely on Reinforcement Learning (RL) which is one of the rare options when the feedback from the system is limited to a reward. Yet, RL is known to be less efficient than supervised learning due to limited information

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 2.7: **Physics-guided training** – (a) The model introduced in Bézenac et al. (2019) leverages physics priors on dynamics of the sea surface temperature. It is composed of a convolution-deconvolution network forecasting a displacement map $\hat{w}(\boldsymbol{x}, t_k)$ from a set of previous temperature maps. This displacement map is used in an advection-diffusion equation to forecast temperatures at the next time step. (b) Visual examples from the model. (figures from Bézenac et al. (2019))

provided by the reward signal. On the other hand, methods based on imitation suffer from exploration challenges and require extensive data (Lee et al., 2015; Seita et al., 2020).

A promising approach involves employing differentiable simulators that allow gradient descent optimization through the simulator. When the dynamics of interest is differentiable, such as in fluid mechanics (Holl et al., 2020; Um et al., 2020), gradients can be straightforwardly calculated through a discrete model, which provides direct feedback on how the action chosen by the agent would have affected the environment. Challenges arise when dealing with non-differentiable dynamics, particularly in simulations involving rigid or soft bodies, since contact forces are not differentiable. Standard approaches consist in smoothing stiff physical laws to create simulators capable of handling rigid (Toussaint et al., 2019; Giftthaler et al., 2017; de Avila Belbute-Peres et al., 2018; Geilinger et al., 2020) or soft dynamics (Ma et al., 2021; Du et al., 2021; Qiao et al., 2020; Hu et al., 2019). These simulators are used for various downstream tasks, including object manipulation (Chen et al., 2023; Li et al., 2023), motion planning (Ren et al., 2023), or estimating physics properties from videos.

While hybrid models possess notable advantages, they still suffer from limitations. Training the residual component introduces challenges, as f_θ typically requires a classical numerical solver, which itself can be nontrivial, in particular for PDEs. Furthermore, if the residual is represented by a neural network, the chosen numerical scheme must support gradient propagation for effective optimization. Unfortunately, this requirement often leads to slow training time.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

2.5 The Return of Physics

The appeal of machine learning methods for dynamical modeling lies in their capacity to learn intricate patterns and representations from data. This becomes especially valuable when the underlying physics of the system is not fully understood or difficult to model.

There is a trade-off to be made though. On one hand, handcrafted models are tedious to build and require accurate experiments to identify the parameters but offer interpretable results, thus predictable behavior. Data-driven techniques, on the other hand, are much more versatile, and easier to use, with minimum requirements in terms of expert knowledge. However, the resulting model is hard to understand, and the more powerful the model the harder it is to derive theoretical guarantees. A promising balance of the advantages of both methods lies in methods based on symbolic regression. These techniques propose to infer a symbolic dynamical model from a large dataset of trajectories gathered on the system.

SINDy (Brunton et al., 2016) SINDy is an algorithm that uncovers governing equations from data. Instead of directly searching for a specific analytical form of the function f_θ, the algorithm constructs a **dictionary** of d candidate terms that may appear in the dynamics

$$\boldsymbol{\Theta}(\boldsymbol{s}) = \begin{bmatrix} | & | & & | & | & | & \\ \boldsymbol{s} & \boldsymbol{s}^2 & \dots & \cos \boldsymbol{s} & \sin \boldsymbol{s} & \exp \boldsymbol{s} & \dots \\ | & | & & | & | & | & \end{bmatrix} \in \mathbb{R}^{n \times d}. \tag{2-38}$$

The dynamical model is then assumed to be a linear combination of the terms in the dictionary, i.e.

$$\dot{\boldsymbol{s}} = \boldsymbol{\Theta}(\boldsymbol{s})\boldsymbol{\Xi}, \tag{2-39}$$

where $\boldsymbol{\Xi} \in \mathbb{R}^{d \times n}$ is a parameter vector to be estimated. To limit the overall complexity of the model, it is assumed that $\boldsymbol{\Xi}$ is sparse, meaning that most of its terms are zero. Solving for sparsity can be achieved by numerous methods, e.g. LASSO (Tibshirani, 1996), or a sequential thresholded least-squares algorithm.

This work has been expanded to include PDEs by incorporating partial derivatives into $\boldsymbol{\Theta}$ (Rudy et al., 2017; Schaeffer, 2017). Furthermore, ensemble methods such as bootstrap aggregation (Breiman, 1996) have been applied in a follow-up work (Fasel et al., 2022). Symbolic regression can also be approached using genetic algorithms (Cranmer, 2023; Schmidt and Lipson, 2009; Searson et al., 2010) combining operators like $+, -, \div, \times, \exp, \log, \dots$ to form a symbolic expression. For high dimensional problems, such as particle physics, an intermediate step is introduced in (Cranmer et al., 2020a), involving fitting a neural network to model particle interaction, followed by symbolic regression to simplify the model. It is also possible to directly regress a symbolic expression from the weights of a neural network, provided that activation functions are expressive enough to capture complex relationships, and under constraints of weight sparsity through L_1 regularization (Sahoo et al., 2018; Long et al., 2019).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Recently, the Monte-Carlo Tree search has been used to explore expression trees and regress symbolic equations (Sun et al., 2023).

Sparse regression of symbolic equations has garnered significant interest in the physics community: while relieving the burden of physics modeling, it still produces human-interpretable and explainable formulas suitable for downstream tasks. However, the method requires that the terms appearing in the true dynamics are included as candidates in the dictionary, necessitating a careful design of $\boldsymbol{\Theta}$.

2.6 Take-home messages

We derived a general framework called *latent dynamics* inspired by the state-space representation of dynamical systems, which is ubiquitous in physics and control theory. This setup is composed of a state-observer (an encoder) to estimate a state variable in a latent space from a set of previous observations, followed by a dynamics function and an observation function. This setup has been declined for various purposes such as enforcing a particular form for the dynamics or ensuring stability properties. However, it also requires hypotheses on the dynamics and the interval of past observations required to retrieve the state, which will be discussed in Part II.

Time-continuous dynamics are usually preferred over discrete systems since more algorithms and laws are tailored to them. Neural-ODEs provide a convenient framework for explicitly leveraging continuous dynamics in a numerical implementation of a latent dynamics. This also allows the incorporation of advanced priors into the model by integrating analytical expressions. Yet, such efforts can prove to be ineffective when the data-driven part becomes more expressive: handcrafted inductive biases have inherent limitations in complexity and may be easily learned by the model. However, hybrid models have shown great potential to satisfy the need for interpretability or to handle low-quality datasets.

Nevertheless, these approaches often come at the expense of increased computational complexity due to multiple passes over the computation graph. Consequently, these methods are currently restricted to relatively simple tasks. To scale up, a shift in the approach is necessary, involving a fresh perspective on intuitive physics that relaxes certain prerequisites. Embracing this new approach enables addressing large-scale datasets and complex problems by harnessing the power of advanced deep learning models to their full potential. This will be the topic of the next chapter.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 3

Deep Learning for intuitive physics

So far, we addressed system identification *stricto sensu* by looking for explicit approximations of the dynamics (f, h), bearing in mind that the learned state-space representation could be used for different purposes later on. This is actually restrictive since the architecture must eventually be related to the latent dynamics setup.

However, system identification is typically an intermediate step toward a higher goal. Sometimes, this step be bypassed to directly tackle the final objective. Doing so usually offers shortcuts and new solutions in which the dynamics is solely implicitly defined and embedded into the weights of a neural network. In this chapter, we study approaches from intuitive physics, which performs by relying on shortcuts and regularities in data rather than explicit dynamical modeling. In particular, we identified two guidelines to alleviate the needs of system identification. These workarounds propose to

- **Learn the solution** – by directly parametrizing the trajectory of a dynamical system rather than its dynamics. In other words, it consists of approximating the solution of a physics problem (typically a PDE) with a neural network $s_\theta(\boldsymbol{x}, t)$. When the PDE operator is known, training the parameters θ can be achieved by taking advantage of automatic differentiation tools. Training is more complicated when the operator remains unknown. Such techniques will be explored in Part III of the manuscript.

- **Learn the solver** – differs from discrete-time system identification since the notion of state is not explicitly required. Instead, we are looking for a function capable of advancing a trajectory forward in time, as a conventional simulator would do. This setup is arguably the most popular in the community and benefits from a large body of work, including ours, presented in Part IV.

Both approaches raise strong interest from the industrial community, as simulation is at the core of many applications. By trying to *learn the solution*, we end up building new solvers for PDE using simpler and more user-friendly tools than off-the-shelf software. Moreover,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

learning the solver offers great opportunities to accelerate physical simulations, reducing their cost and facilitating their usage.

3.1 Learning the solution from PDE operator

Among the vast landscape of tasks tackled by deep learning, the ones we are going to introduce belong to the few who have successfully crossed the frontier and are now researched from both ends by physicists and data scientists. Indeed, a large part of the work in applied physics and engineering is dedicated to system modeling, which often requires a numerical solver, either to identify parameters of the model or simply to ensure that its predictions match the observations. On one hand, for some problems, the solver can be relatively easily implemented in a few lines of code by a non-expert[1] (finite difference method for instance). On the other hand, more complex physics, such as fluid mechanics or multi-dimensional systems, requires carefully designed algorithms and generic simulation software may fall short of flexibility for research purposes.

Physics-Informed Neural Networks (PINNs) are a promising family of methods for building a solver which, in theory, can be applied to any PDE-based problem with minimal modifications, while remaining simple to implement. The technique has rapidly harnessed interest from other fields and is presently used in several research areas that were not related to Artificial Intelligence (AI) before. This line of work assumes that the dynamical model is already known, and takes the form of a PDE

$$\begin{cases} \dot{\boldsymbol{s}}(\boldsymbol{x},t) &= f(\boldsymbol{s},t,\boldsymbol{x},\nabla\boldsymbol{s},...,\nabla^{(n)}\boldsymbol{s}) \quad \forall \boldsymbol{x},t \in \Omega\times\mathcal{T} \\ \boldsymbol{s}(\boldsymbol{x},t) &= g(\boldsymbol{x},t) \qquad\qquad\qquad\qquad\quad \forall \boldsymbol{x},t \in \partial\Omega\times\mathcal{T} \end{cases}, \tag{3-1}$$

where $\partial\Omega$ is the frontier of the spatial domain.

PINNs model the solution of equation 3-1 with a neural network $\boldsymbol{s}_\theta(\boldsymbol{x},t)$, and learn the parameters in order to satisfy the PDE operator, starting from a known initial condition $\boldsymbol{s}(\boldsymbol{x},0)$. Deep learning and PDEs have a long shared history (Dissanayake and Phan-Thien, 1994; Lagaris et al., 1998; Psichogios and Ungar, 1992) and have recently regained attention (Raissi et al., 2019, 2017). Unlike conventional solvers, PINNs stand out for their *mesh-free* nature: the solution is not modeled on a finite subset of $\Omega\times\mathcal{T}$, but can rather be evaluated at arbitrary points within the training domain. However, note that PINNs are often trained on a bounded set $\mathcal{T} = [0,T]$ thus extrapolation to points outside the training domain is not granted. The choice of architecture for $\boldsymbol{s}_\theta$ is left free. Commonly used architectures include MLPs, but more recent approaches have been proposed, such as sinus-activated neural networks (Sitzmann et al., 2020) or Multiplicative Filters Networks (Fathony et al., 2021).

A typical loss for training a PINN exploits the residual given by the PDE operator. The

[1] Probably in MatLab though...

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

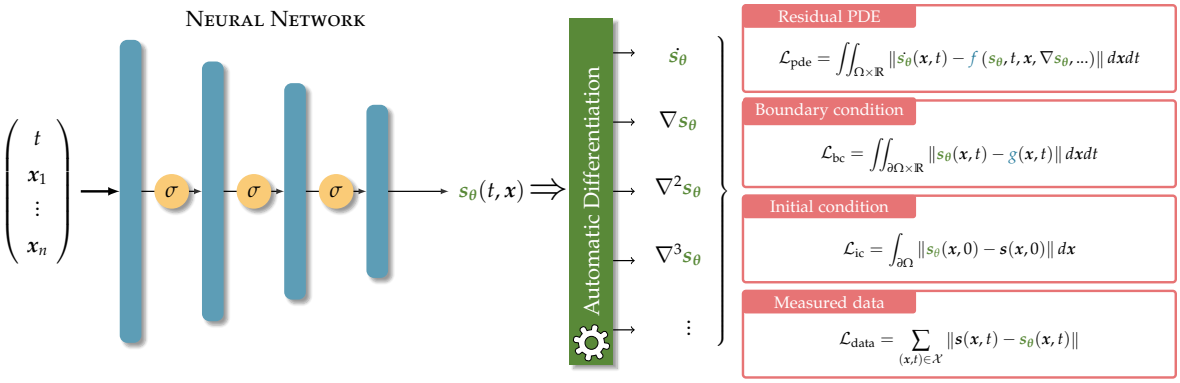


Figure 3.1: **Physics-Informed Neural Networks** – model the solution of a partial differential equation using a deep neural network that takes spatial and temporal coordinates as input and produces the corresponding solution as output. To ensure the physics, boundary and initial conditions are satisfied, these conditions are incorporated as soft constraints within the objective function. Efficient computation of the derivative, necessary for evaluating the residual, can be achieved through the use of automatic differentiation tools. (figure inspired from Cuomo et al. (2022))

technique takes advantage of tools for automatic differentiation of neural networks, which allows to efficiently compute the exact analytical partial derivatives of s_θ. The training objective may encompass the following terms:

1. **Residual on PDE** – This loss computes the difference between the partial derivatives of s_θ and the desired value according to the operator

$$\mathcal{L}_{\text{pde}} = \iint_{\Omega\times\mathcal{T}} \left|\dot{s}_\theta(x,t) - f\left(s_\theta, t, x, \nabla s_\theta, \cdots\right)\right| dxdt, \tag{3-2}$$

   for some norm $|\cdot|$ where the derivative $\dot{s}_\theta$ is calculated analytically through automatic differentiation. Note that the integral form is not suitable for practical implementation and needs to be discretized by sampling the domain $\Omega \times \mathcal{T}$. Originally, latin hypercube sampling was used (Raissi et al., 2019), yet most applications use uniform sampling. Adaptive variants exist to increase accuracy near regions with lower accuracy (see Wu et al. (2023) for a review).

2. **Boundaries and initial condition** – The traditional formulation of PINNs relaxes the enforcement of boundary and initial conditions with soft constraints, which are regularized by another loss term:

$$\mathcal{L}_{\text{boundary}} = \underbrace{\iint_{\partial\Omega\times\mathcal{T}} \left|s_\theta(x,t) - g(x,t)\right| dxdt}_{\text{boundary condition}} + \overbrace{\int_{\Omega} \left|s_\theta(x,0) - s(x,0)\right| dx}^{\text{initial condition}}. \tag{3-3}$$

   Imposing boundary conditions has proven to be of key importance for training PINNs (Wang et al., 2021; Dwivedi and Srinivasan, 2020). Techniques to maintain hard constraints have been studied in Lagaris et al. (2000), and other work proposes to use a

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

distance metric from the boundary to manually enforce the constraints (Berg and Nyström, 2018; McFall and Mahan, 2009; Sheng and Yang, 2021). Recently, Hao et al. (2023) proposed a bi-level optimization procedure where a PINN is used to solve the constraints during an inner loop optimization. The outer loop uses a Broyden method to approximate the gradients.

3. **Observed data** – when available, can be incorporated in an additional term. This tends to prevent over-fitting on small datasets and ensures that the solution aligns with the measurements. The data loss term measures the difference between $s_\theta(x,t)$ and the observed data $s(x,t)$ at sparse measurement points in time and space $\mathcal{X} = \{x_k, t_k) | k \in [\![0, K]\!]\}$:

$$\mathcal{L}_{\text{data}} = \sum_{k=0}^{K} \left| s(x_k, t_k) - s_\theta(x_k, t_k) \right| \tag{3-4}$$

For a general introduction to PINNs, Cuomo et al. (2022) provides an excellent resource. Specific applications of PINNs to fluid dynamics can be found in Cai et al. (2021a), while Cai et al. (2021b) explores their applications in heat transfer and Misyris et al. (2020) focuses on power systems. The literature also offers a large set of training tricks to improve accuracy or convergence speed. In addition to the adaptive collocation point methods discussed above, one can also use adaptive weighting of each loss term (Meer et al., 2022; McClenny and Braga-Neto, 2020). It is also recommended to carefully design the training curriculum of PINNs when addressing challenging PDEs (Wang et al., 2022a; Wight and Zhao, 2020; Krishnapriyan et al., 2021).

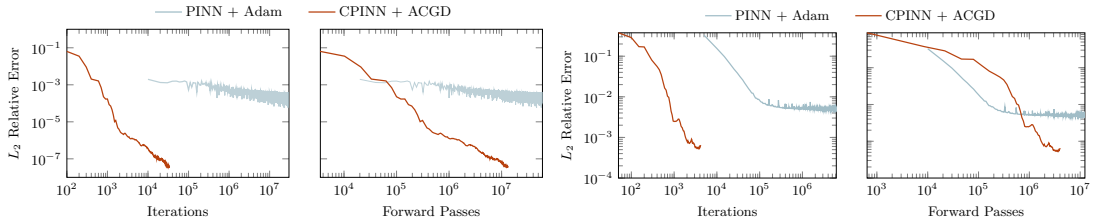
For instance, Zeng et al. (2023) introduces an adversarial discriminator network $c_\theta(x,t)$ to place bets on whether s_θ will overshoot or undershoot the residual at a given location. This network is jointly trained with the solution to minimize:

$$\max_{c_\theta} \min_{s_\theta} \sum_{n=0}^{N} c_\theta(x_n, t_n) \times \left(\dot{s}_\theta(x_n, t_n) - f(x_n, t_n, s_\theta, \nabla s_\theta, \cdots) \right) \tag{3-5}$$

over a set of N evaluation points $(x_n, t_n)_{n=0..N} \in \Omega \times \mathcal{T}$. This formulation defines a minimax game with the Nash equilibrium at $c_\theta = 0$ and $(s_\theta - f) = 0$. The neural networks can be trained using adaptive competitive gradient descent. Note that the norm in the classic PINN loss (equation (3-2)) has disappeared. This is motivated by numerical analysis of the ill-conditionness of the problem implied by the L_2 norm. This multi-agent framework for training PINNs shows substantial gain in performance (see figure 3.2). Other methods avoid the squared loss by relying on the weak formulation of the PDE (E and Yu, 2017; Liao and Ming, 2019), or on game theory (Zang et al., 2020).

Yet, PINNs have many drawbacks and limitations, which impact their applicability to many PDEs, and make their usage and training difficult. In particular:

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) Poisson equation
$\Delta s(x,y) = -2\sin(x)\cos(y)$

(b) Schrödinger equation
$\dot{s} + \frac{1}{2}\frac{\partial^2 s}{\partial x^2} + |s|^2 s = 0$

Figure 3.2: **Competitive PINN** – comparison of the relative residual error for a standard PINN model optimized with Adam algorithm and a Competitive PINN (Zeng et al., 2023) trained with adaptive competitive gradient descent. For both tasks, PINN reach a plateau in accuracy, while CPINN continues to improve beyond this limit.

- A solution obtained with a PINN is **limited to a single initial condition**. To simulate a trajectory from another starting point or different boundary conditions, the model must be trained again from scratch.
- PINNs require the **PDE operator to be known**. Consequently, they are limited to known and modeled phenomena and cannot retrieve a dynamics from a dataset of observations.
- They also suffer from a well-documented pathology: the optimization of the residual from the PDE operator is an **ill-posed problem**, making the neural network particularly difficult to train on non-trivial equations (Krishnapriyan et al., 2021; Wang et al., 2022b, 2021; Liu et al., 2021).

In summary, PINNs offer simplicity and adaptability for solving PDEs, making them accessible to beginners and attractive to physicists. However, their formulation is ill-posed, with important training challenges. Moreover, PINNs requires knowing the PDE, and their main limitation is the inability to generalize to new initial conditions. Rapid adaptation is hindered as it necessitates re-training the neural network from scratch, limiting its use in real-time applications such as trajectory planning.

3.2 Learning the solution from sparse observations

Fortunately, new techniques are emerging to maintain the advantages of PINNs (mesh-free, dense inputs) without their drawbacks (knowledge of the PDE, generalization to new initial conditions).

3.2.1 *Neural operators*

A possible approach consists in learning a mapping from a functional space to another, also called *operator* $\mathcal{G}_\theta : \boldsymbol{u} \rightarrow \boldsymbol{s}$. When applied to PDEs, the input function $\boldsymbol{u}$ can be for instance a new initial condition, different physical parameters, or a control input signal. In theory, since

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

the output of $\mathcal{G}_\theta$ is also a function, we obtain a mesh-free solution capable of generalizing to different inputs $\boldsymbol{u}$. In practice, the input and output functional spaces need to be sampled for numerical implementation, resulting in a training dataset of functions $\mathcal{D} : \left\{ (\boldsymbol{u}_i, \boldsymbol{s}_i) | i \in [\![0, N]\!] \right\}$ evaluated at finite discrete points $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\} \subset \Omega$ (for readability, time and space variables have been aggregated in a single vector $\boldsymbol{x}$). To be mesh-free, the neural operator must be able to interpolate its output at any location $\boldsymbol{x} \in \Omega$, potentially outside the training domain $\mathcal{X}$.

DeepONet (Lu et al., 2019) is a neural operator composed of two modules. First, given an input function $\boldsymbol{u}$ sampled at fixed points in time and space $\mathcal{X}$, the *branch network* computes K representations $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_K$ from each value $\boldsymbol{u}(\boldsymbol{x}_1), \cdots, \boldsymbol{u}(\boldsymbol{x}_K)$. These embeddings can be computed jointly with a single neural network, or separately, with a different neural network for each input $\boldsymbol{u}(\boldsymbol{x}_k)$.

The output function of the neural operator is the solution of the PDE $\boldsymbol{s}(\boldsymbol{x}_q)$ expressed at an arbitrary query point $\boldsymbol{x}_q \in \Omega$. In DeepONet, the output is assumed to be a linear combination of $\boldsymbol{b}_k$ with weights coefficients $w_1, \cdots, w_K$ depending on the query $\boldsymbol{x}_q$. These coefficients are computed with a *trunk network*:

$$\boldsymbol{s}(\boldsymbol{x}_q) = (G\boldsymbol{u})(\boldsymbol{x}_q) \approx \sum_{k=1}^{K} w_k \boldsymbol{b}_k \tag{3-6}$$

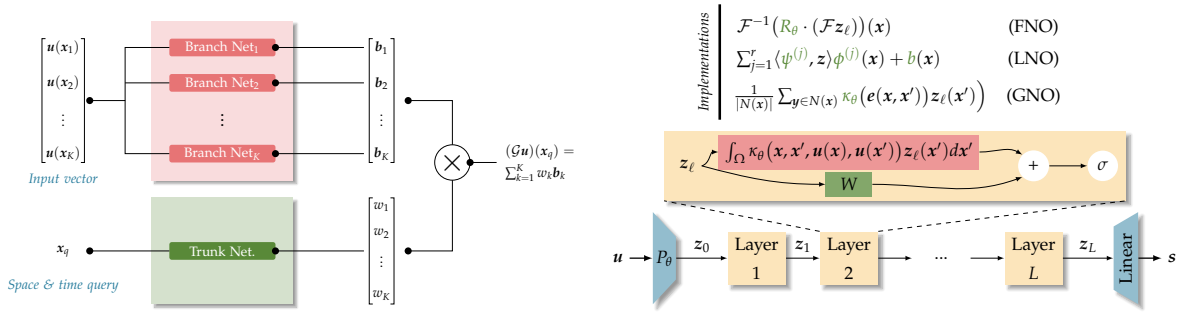
DeepONet (illustrated in figure 3.3a) addresses some limitations of PINNs. It generalizes to unobserved scenarios during training, and offers dense inputs in Ω, while not requiring the analytic PDE during training. Lu et al. (2019) also provides theoretical guarantees regarding approximation error, considering assumptions about the density of the observation grid. Yet, contrary to PINN, the model is not entirely mesh-free as the input function $\boldsymbol{u}$ must be discretized on a set of static points in time and space $\mathcal{X}$ that cannot change during evaluation.

Another line of work takes inspiration from the variational form of the PDE and proposes to derive neural operators as a multi-layer network based on the following equation:

$$\boldsymbol{z}_{\ell+1}(\boldsymbol{x}) = \sigma \left(W\boldsymbol{z}_\ell(\boldsymbol{x}) + \int_\Omega \underbrace{\kappa_\theta(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{u}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}'))}_{\text{Kernel integral operator}} \boldsymbol{z}_\ell(\boldsymbol{x}') d\boldsymbol{x}' \right), \tag{3-7}$$

where $\boldsymbol{z}_\ell$ is the output of the ℓth layer of the neural operator, σ is an activation function and W, θ are learnable parameters. The *kernel integral operator* is the most important part of the architecture and is inspired by the Green function of PDEs. In other words, each layer behaves like the solution of a PDE implicitly defined by the kernel operator. This function can be modeled with various strategies to replace the integral form with a discrete formulation.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 3.3: **DeepONet & Neural Operators** – (a) DeepONet consists of a *branch network* that outputs b_k (representing the solution to the PDE) at fixed points in time and space x_k, either with a single MLP or a set of networks. The *trunk network* calculates interpolation weights w_k to express the solution at arbitrary query points x_q (figure inspired from Lu et al. (2019)). (b) Neural Operators learn operators mapping between functional spaces. The input, representing the initial condition or command signal, is projected to a latent space and undergoes kernel transformation to approximate the PDE solution. Several methods are available to approximate the kernel operation, such as Graph Neural Operator (GNO), Low-rank Neural Operator (LNO), or Fourier Neural Operator (FNO) (figure inspired from Kovachki et al. (2021)).

For instance, given a set of measurement points $\mathcal{X}$, (Li et al., 2020b) approximates the kernel using local interactions between points. In other words, for any point $x \in \mathcal{X}$, the kernel is assumed to be non-zero only for points $y \in \mathcal{X}$ in the neighborhood of x, noted $\mathcal{N}(x)$. First, the input function $u(\mathcal{X})$ is smoothed by adding Gaussian noise, and projected into an embedding space z_0 thanks to a neural network P_θ acting as an encoder (typically an MLP). It then undergoes several layers of transformations given by

$$\begin{aligned} &\underline{\text{Initialisation:}} && z_0(x) && = P_\theta\big(x, u(x), \overbrace{u_\epsilon(x), \nabla u_\epsilon(x)}^{\text{Gaussian smoothed}}\big) \\ &\underline{\text{Iterative update:}} && z_{\ell+1}(x) && = \sigma\Big[W z_\ell + \tfrac{1}{|\mathcal{N}(x)|}\textstyle\sum_{x'\in\mathcal{N}(x)} \kappa_\theta\big(x, x, u(x), u(x')\big) z_\ell(x')\Big], \\ &\underline{\text{Output:}} && s(x) && = Q z_L + q, \end{aligned} \tag{3-8}$$

where L is the number of layers, $|\mathcal{N}(x)|$ is the cardinal of the neighborhood of x, and W, Q, q and θ are trainable parameters. The kernel is said to be local since the output $z_\ell(x)$ only depends on the previous vector $z_{\ell-1}$ evaluated on the neighborhood of x. The model is called *Graph Neural Operator* (GNO) and is illustrated in figure 3.3b.

Graph neural operators benefited from several follow-up works such as *Low-rank Neural Operator* (LNO) (Kovachki et al., 2021) which reduces computational complexity using tensor products, and *multi-pole graph neural operator* (Li et al., 2020c), which processes the initial graph at multiple coarseness levels. A markovian variant is introduced in Li et al. (2021) (MNO) and a boundary enforcement technique (Saad et al., 2022) has also been introduced. The *Fourier Neural Operator* (FNO) (Li et al., 2020d) is a popular choice, utilizing the Fourier transform to

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

compute the kernel operator.

$$\int_{\Omega} \kappa_\theta\big(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}')\big) \boldsymbol{z}_\ell(\boldsymbol{x}') d\boldsymbol{x}' = \mathcal{F}^{-1}\big(R_\Phi \cdot (\mathcal{F}\boldsymbol{z}_\ell)\big)(\boldsymbol{x}), \tag{3-9}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are respectively Fourier transform and inverse Fourier transform. R_Φ is the only matrix parameter. However, working with non-uniform meshes poses challenges due to the discretization of the Fourier transform.

Yet, while it is true that neural operators (equation (3-7)) can theoretically generalize to unseen initial conditions and arbitrary locations, this is actually not the case in practice. For instance, FNO requires a static cartesian grid and cannot be directly evaluated outside $\mathcal{X}$. Similarly, GNO can handle arbitrary meshes in theory but still has limitations in evaluating points outside the training grid and the MNO variant can only be queried at fixed time increments. Our quest for an effective framework to learn a solution capable of generalizing to new initial condition and handling mesh-free data is not over yet, and we will provide a contribution in chapter 7.

3.2.2 *Input-continuous and generalizable approaches*

An interesting analogy can be made between PINNs and Implicit Neural Representations (INRs), such as Neural Radiance Fields (NeRF (Mildenhall et al., 2020)), where objects (images, videos, or signed-distance functions of 3D objects) are directly encoded into the weights of a neural network. Similarly, a PINN encodes a unique solution within a neural network, hence the limitation to a single initial condition. To achieve generalizable PINNs, there are probably insights to be harnessed from INRs.

Generalization in an INR can be simply achieved with the help of a feature code, concatenated to the inputs. This trick consists of assigning a feature code to each object in the dataset while keeping fixed weights within the model. This feature code is used by the neural network to change its output accordingly (Mescheder et al., 2019; Park et al., 2019). Recent work shows that a better approach is to apply dynamic *weight shifting and scaling* using modulation vectors (Dupont et al., 2022a,b; Mehta et al., 2021a). In that case, the feature code is not concatenated to the input but rather used to scale the weights of each layer.

Dynamics-aware Implicit Neural representations (DINo) (Yin et al., 2022) takes advantage of this technique to achieve generalization over new initial conditions. The model proposes to learn a spatial INR using an adapted Multiplicative Filter Network $\boldsymbol{s}(\boldsymbol{x}, t) = h_\theta\big(\boldsymbol{x}, \boldsymbol{\alpha}(t)\big)$ (Fathony et al., 2021) and modulates its weights dynamically across time with a modulation

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

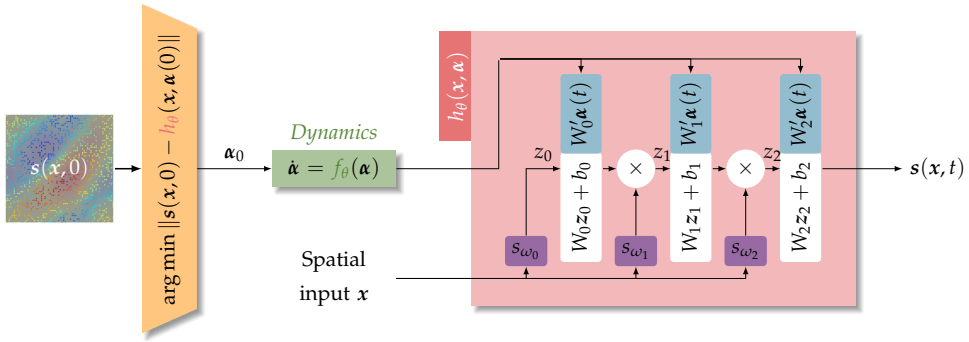


Figure 3.4: **DINo** – utilizes an implicit representation $h_\theta(x,\alpha)$ to represent snapshots of the solution by modulating its weights with a code vector α. This code follows a Neural-ODE providing the time dependency of the solution. The initial condition $\alpha(0)$ is obtained via auto-decoding from the initial state $s(x,0)$ (figure inspired from Yin et al. (2022)).

vector $\alpha(t)$:

$$h_\theta(x,\alpha(t)) = z_L(x) \text{ with } \begin{cases} z_0(x) &= s_{\omega_0}(x) \\ z_\ell(x) &= \left(W_{\ell-1}z_{\ell-1}(x) + b_{\ell-1} + \overbrace{W'_{\ell-1}\alpha(t)}^{\text{modulation}}\right) \odot s_{\omega_\ell}(x) \\ z_L(x) &= W_{L-1}z_{L-1} + b_{L-1} \end{cases}, \quad (3\text{-}10)$$

where $s_{\omega_\ell}(x) = [\cos(\omega_\ell x)\ \sin(\omega_\ell x)]^T$ is a Fourier basis for the layer ℓ with trainable parameters ω_ℓ and z_ℓ is the latent state of the ℓth layer. The model leverages an additional term $W'_{\ell-1}\alpha(t)$ depending on time which modulates the output of each layer. The temporal evolution of the solution is thus handled by this modulation vector, which is assumed to obey an ODE:

$$\dot{\alpha}(t) = f_\theta(\alpha), \qquad \alpha(t{=}0) = e_\theta(s(x,0)), \quad (3\text{-}11)$$

The dynamic f_θ is modeled by an MLP embedded in a Neural-ODE setup. The encoder is an optimization process $\alpha(0) = \arg\min \|s(x,0) - h_\theta(x,\alpha(0))\|$. The model is depicted in figure 3.4.

DINo has quickly captured the attention of the research community due to its resemblance to PINNs while overcoming some of their limitations. However, it also requires underlying hypotheses for this decomposition to be possible. It assumes the existence of an ODE which represents the evolution of α. In other words, it assumes that the state dynamic of the modeled system translates into a suitable dynamic in the parameter space of the implicit representation, which itself needs to be predictable by a latent space governed by an ODE. Moreover, these transformations should be sufficiently simple (in a functional sense) to be trainable end-to-end.

Another way of approaching the task consists of decoupling the prediction task given the initial condition from the interpolation at arbitrary query points. This approach builds upon auto-regressive solvers, which will be discussed more thoroughly in the next section. This type of model advances a state representation one step forward in time but is limited in fixed

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

location and timesteps. A natural solution, while not straightforward to achieve, is to append a module to interpolate the forecasted solution at arbitrary locations.

Mesh-Agnostic Neural PDE Solver or MAgNet (Boussif et al., 2022) uses an *encode-interpolate-forecast* framework. The model performs by interpolating the observed initial condition to the queried positions and forecasts the solution using an auto-regressive solver. In particular, the interpolation is performed in a latent space $z(\mathcal{X})$, with $\mathcal{X} = \{x_k | i \in [\![0, K]\!]\}$, rather than in the physical space $s(\mathcal{X}, 0)$ defined by the initial condition.

$$z(x_k) = P_\theta\big(s(x_k, 0)\big). \tag{3-12}$$

The interpolation at spatial query points $\mathcal{X}' = \{x'_j | j \in [\![0, L]\!]\}$ is performed using a learned projection function (Esmaeilzadeh et al., 2020; Chen et al., 2021c) operating in a latent space defined by the vector z. The physical value is retrieved using a decoder (MLP):

$$z(x'_j) = \frac{\sum_{i \in \mathcal{N}(x'_j)} w_j g_\theta\big(s(x_i), z(x_i), x_i - x'_j\big)}{\sum_{i \in \mathcal{N}(x'_j)} w_j}, \tag{3-13}$$

$$\hat{s}(x'_j, 0) = d_\theta\big(z(x'_j)\big), \tag{3-14}$$

with $\mathcal{N}(x'_j) \subset \mathcal{X}$ the neighborhood of x'_j, w_j are distance-based weights, and g_θ, d_θ are MLPs. We obtain a new initial condition $\hat{s}(\mathcal{X} + \mathcal{X}', 0)$ augmented with values at the query points, which can be advanced in time using an auto-regressive forecaster F_θ. In particular, MaGNet uses an Euler scheme with a timestep Δt between each iteration:

$$\hat{s}(x'_j, t + \Delta t) = s(x'_j, t) + \Delta t \times F_\theta\big(\hat{s}(\mathcal{X} + \mathcal{X}', t)\big) \tag{3-15}$$

Combining auto-regressive prediction and interpolation has great potential for achieving space and time continuous simulation: learning an input-dense solution capable of generalization. Yet, MaGNet suffers from several flaws that hinder its performance. The interpolation is performed solely on the initial condition, ignoring insights that could be gathered from multiple timesteps. Moreover, if the number of query points exceeds the number of known points ($|\mathcal{Y}| \gg |\mathcal{X}|$), the input to the auto-regressive solver is filled with noisy interpolation, which impacts performance. We propose a different approach in Chapter 7, where interpolation is performed after forecasting, and which leverages information from both spatial and temporal domains.

3.3 Learning the solver for grid-based data

So far, we have primarily focused on dense representations of trajectories from physical systems, which find usage in experimental sciences where data is often scattered across time and space. However, for simulation purposes, a discrete representation of the solution often suffices, that is measurements taken at regular time intervals $\mathcal{T}=t_1, ..., t_K$ and fixed positions $\mathcal{X}=x_1, ..., x_M$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

In this section, we will focus on approaches trying to learn the behavior of a solver directly, without explicit modeling of the dynamics. We recall that this guideline differs from learning a discrete dynamical system, in the sense that the notion of state is not necessarily required. In other words, we adopt a regression equation in a discrete-time and space setup:

$$\boldsymbol{s}(\mathcal{X}, t_{k+1}) = F\big(\boldsymbol{s}(\mathcal{X}, t_k)\big), \tag{3-16}$$

where F is the so-called *solver*. Thus, a trajectory can be forecasted from an initial condition by applying F to its output multiple times. Our goal is to approximate F from a dataset $\mathcal{D}$ of trajectories sampled at fixed points $\mathcal{X} \times \mathcal{T}$.

3.3.1 *Grid-based simulations*

Let us begin with the particular case where $\mathcal{X}$ is a uniform and regular grid in a 2D space. This discretization scheme is common for several applications involving PDEs. The interest we have in this setup is that it is highly related to a well-studied field of deep learning: computer vision. Indeed, assuming that the measurement points are distributed on a uniform and regular grid allows to interpret trajectories as videos, expressing the physics forecasting into a video prediction task which has been successfully addressed with CNNs (Wang et al., 2022d; Lee et al., 2018; Gao et al., 2022; Wang et al., 2019e).

For instance, fluid dynamics is governed by the Navier-Stokes equations, which can be written in the incompressible case and without heat transfer as:

$$\nabla \cdot \boldsymbol{v} = 0; \quad \rho\Big(\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v}\Big) = -\nabla p + \mu \Delta \boldsymbol{v} + \boldsymbol{f}, \tag{3-17}$$

where $\boldsymbol{v}(\boldsymbol{x}, t), p(\boldsymbol{x}, t)$ are the velocity and pressure field, ρ the density, μ the viscosity coefficient and $\boldsymbol{f}$ some external perturbations. Simulating fluid dynamics is challenging, in particular because of the absence of an independent equation for pressure. Typically, the pressure field is deduced from correction schemes to ensure incompressibility (Issa, 1986; Barton, 1998; Kim and Benson, 1992). For 2D flows, the spatial domain can be discretized in a uniform grid, which has the advantage of simplifying the discretization scheme, allowing to use finite differences. However, the method is known to be inaccurate for complex flows (Versteeg and Malalasekera, 2007).

Conversely, a learning-based approaches are a convenient solution to bypass the limitations of handcrafted algorithms by discovering regularities and shortcuts in data, thus relying on intuitive physics. In this case, CNNs are a well-adapted structure for modeling the dynamics (F). Indeed, such model can be trained to simulate the trajectory of the system forward in time in a much more complex way than the simpler finite difference method. Interestingly, operators of spatial differentiation on a 2D grid can be expressed with convolution kernel, aligning CNNs with conventional solvers.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) Ground Truth (b) DeepFluid

Figure 3.5: **DeepFluid**: Simulation results from Kim et al. (2019), where a CNN-based model is used to perform fluid simulations for computer graphics. The model is trained on simulations obtained with Mantaflow (Pfaff and Thuerey, 2016), a simulation software inspired by the Stable Fluid algorithm (Stam, 1999), a fast and stable yet not physically accurate algorithm for rendering fluids.

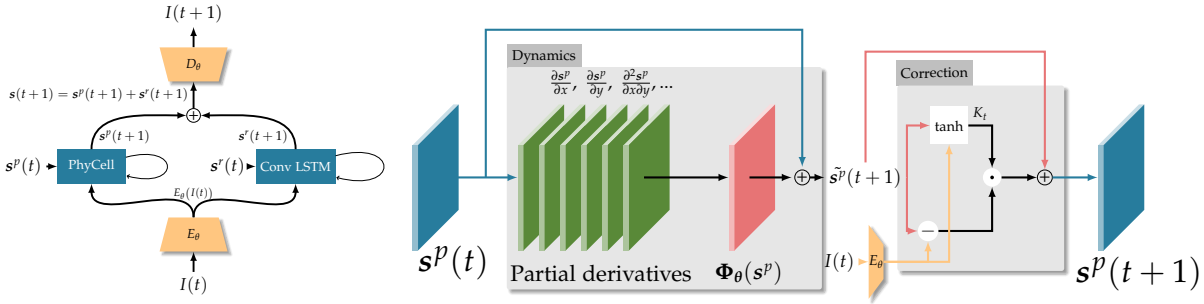
DilResNet (Stachenfeld et al., 2021) models the solver with a CNN-based neural network called *Dilated Residual Network* F_θ. The model is applied recursively from the initial condition to forecast the pressure and velocity fields. The core of the model operates in a latent space $z(\mathcal{X}, t_k) = e_\theta\big(v(\mathcal{X}, t_k), p(\mathcal{X}, t_k)\big)$. Finally, the output is then decoded to obtain the pressure and velocity fields for the next time step.

$$\begin{bmatrix} v(\mathcal{X}, t_{k+1}) \\ p(\mathcal{X}, t_{k+1}) \end{bmatrix} = \begin{bmatrix} v(\mathcal{X}, t_k) \\ p(\mathcal{X}, t_k) \end{bmatrix} + F_\theta\big(z(\mathcal{X}, t_k)\big) \tag{3-18}$$

Due to their early-age popularity in computer vision, CNNs are commonly used to address fluid mechanics simulations. Such auto-regressive simulators for fluid mechanics can be designed either end-to-end (Kurz et al., 2023; Margenberg et al., 2022; Gao et al., 2021; Obiols-Sales et al., 2020; Franz et al., 2021) or embedded in hybrid solvers (Tompson et al., 2017; Kochkov et al., 2021). Training usually involves supervised learning from accurate trajectories from classic solvers or real-world measurements from particle image velocimetry (Eckert et al., 2019; Yu et al., 2023). CNN-based simulators excel at capturing intuitive fluid dynamics and learning shortcuts to accelerate simulation and rendering, and have many applications in computer graphics (Kim et al., 2019; Wiewel et al., 2019) (see figure 3.5). Yet, suitable dataset for learning fluid dynamics are challenging to produce (see chapter 8).

Learning Fluid Mechanics from Scratch (Wandel et al., 2021) proposes a different training technique by leveraging the knowledge of the Navier-Stokes equations. The approach considers a *pool* of simulation results, initialized with random initial conditions. During training, inputs are drawn from the pool and inputted to the auto-regressive model trained to minimize the residual from the Navier-Stokes equation (in a PINN-like setup). After several optimization steps on the weights of the model, the predicted output is added to the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 3.6: **PhyDNet** – is an auto-regressive model forecasting the temporal evolution of videos representing physical phenomena. (a) the model disentangles a physics representation s^p from the residual dynamics s^r, each term having its own dynamics. (b) the physics state is processed by a **PhyCell**, a neural unit based on partial differential equations and inspired by Kalman filtering to forecast and correct the latent state based on the observed frame (figures adapted from Guen and Thome (2020)).

pool of simulations, and the operation is repeated. The pool grows iteratively with better and better outcomes as the model progresses.

The main limitation of CNN-based fluid solvers is their requirement for regular and uniform grids, which is unpractical in computational fluid dynamics. Complex geometries and adaptive resolutions are better handled with irregular meshes (see 8).

However, CNNs are highly relevant for cases where the physical state is measured via a video. Indeed, the *observation function* h of a camera (mapping states to frames) is challenging to model using traditional tools, and CNNs offer a convenient way to learn state representations while avoiding the need for hand-crafted feature extractors. Yet, these models face challenges since videos contain many irrelevant information for physics forecasting, such as colors and backgrounds. Useful information is deeply entangled in the image structure, requiring a holistic analysis of the frame to extract physical properties. Thus, a key task in physical video processing is disentangling dynamic information from visual information, as the latter does not influence the dynamics (Ehrhardt et al., 2018; Jaques et al., 2020).

PhyDNet (Guen and Thome, 2020) achieves this disentanglement by assuming the existance of a latent space in which the evolution of a video $I[k]$ can be decomposed into two components $s[k] = s^p[k] + s^r[k]$, modeled as discretized variables of continuous states $s^p(t)$ and $s^r(t)$ for some time t ($s^p[k] = s(t_k)$). The first component s^p represents the **physical** dynamics, and s^r the **residual** dynamics. Both follow a different ODE in continuous time:

$$\frac{\partial s^p}{\partial t} = F_\theta^p(s^p, I) \qquad \frac{\partial s^r}{\partial t} = F_\theta^r(s^r, I) \tag{3-19}$$

The residual dynamics F_θ^r is modeled with a variant of RNN called ConvLSTM (Shi et al., 2015). For the physical dynamics F_θ^p, PhyDNet introduces a specialized unit called *PhyCell*,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

designed to model PDEs in the latent space. PhyCells behave as a Kalman filter, fusing the prediction from a learned dynamical system and information from the current measurement (i.e. the current image)

$$\begin{aligned} \underline{\text{Dynamics:}} \quad & \tilde{s^p}[k+1] &&= s^p[k] + \Phi_\theta\big(s^p[k]\big) \\ \underline{\text{Correction:}} \quad & s^p[k+1] &&= \tilde{s^p}[k+1] + K[k] \odot \Big(E_\theta(I[k]) - \tilde{s^p}[k+1]\Big) \end{aligned} \tag{3-20}$$

The matrix $K[k]$ is similar to a Kalman gain and is modeled with a RNN $K[k] = \tanh(W_1 \tilde{s^p}[k+1] + W_2 E_\theta\big(I[k]\big) + \boldsymbol{b})$. E_θ is a CNN-based neural network playing the role of a state estimator from the video. The learned dynamics Φ_θ is modeled as follows:

$$\Phi_\theta(s^p) = \sum_{i,j \leqslant q} c_i \frac{\partial^{i+j} s^p}{\partial x^i \partial y^j} \tag{3-21}$$

with $[x\ y]^\top$ are the spatial coordinates within the image and c_i are learnable weights. The function Φ_θ uses convolutional layers to represent differentiation filters. PhyDNet achieves excellent video prediction performance (see figure 3.6 for an overview). The next frame is retrieved with a decoder network $I[k+1] = D_\theta\big(s^r[k+1] + s^p[k+1]\big)$

PhyDNet addresses an "output prediction" task which consists of simulating the future evolution of a measured quantity (here, the frames of a video) over time. This setup is very common, and dedicated models typically exhibit two behaviors:

- **A closed-loop phase** – where ground truth observations $I[k]$ are provided to the model to correct the estimate of the latent state.
- **An open-loop phase** – where the model becomes autonomous, using its own predictions as new inputs.

This raises questions about the amount of previous observations needed during the closed-loop phase to accurately estimate a good latent representation (this question is addressed in chapter 4), as well as the ability of the model to robustly converge to the true state from observed frames (discussed in chapter 5).

Surprisingly, this class of neural simulators requires initialization from another simulator and cannot perform cold start forecasting from a single initial condition. In other words, it requires a window of past observations to perform. These initialization frames are usually available for online tasks (e.g. control or state estimation), but not for simulation purposes. The neural simulators we will introduce in Part IV are capable of forecasting a trajectory from a single initial condition.

3.3.2 *Object-level representations for videos*

In many applications, reasoning over pixels is impractical and unnatural, especially for problems such as rigid body dynamics, where interpreting object movement directly from pixel

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

is challenging. In that case, disentanglement in the sense of PhyDNet might not be adapted. Instead, inspired by and closely linked to computer vision, a line of work suggests reasoning over object-level representations, i.e. structured latent spaces containing more or less explicit information about objects in the scene. This approach benefits from extensive research on object and pose detection (Zhu et al., 2014; He et al., 2017; Carion et al., 2020; Redmon and Farhadi, 2018) and finds usage in visual reasoning tasks (Baradel et al., 2018; Kervadec et al., 2021; Locatello et al., 2020; Xu et al., 2019).

For instance, in rigid body dynamics, one can leverage priors on object behavior, since pixels belonging to the same object should evolve in the same way. Instead of searching for image-wide displacement (i.e. optical flow), pixels are processed in groups, using a unique transformation in 3D space, and deduce how the images should evolve accordingly.

The idea is illustrated in SE3-Net (Byravan and Fox, 2017), which considers a rigid body scene acquired using a depth camera. The network identifies pixels belonging to the same object and estimates their 3D transformations (represented by matrices in $SE(3)$). The next frame of the video is obtained by applying these transformations to the pixels.

Formally, the input frame $I[k]$ at time k is decomposed into Q **motion masks** M^q, where $(M^q)_{i,j}[k]$ represents the probability of pixel (i,j) to belong to the motion class q. The rigid $SE(3)$ transformations $(R_q[k], \boldsymbol{p}_q[k])$ of each motion class are estimated with another network, R_q being a rotation matrix and $\boldsymbol{p}_q$ a translation vector. Each pixel at time k is then displaced to its next position at time $k+1$ using the weighted transformation:

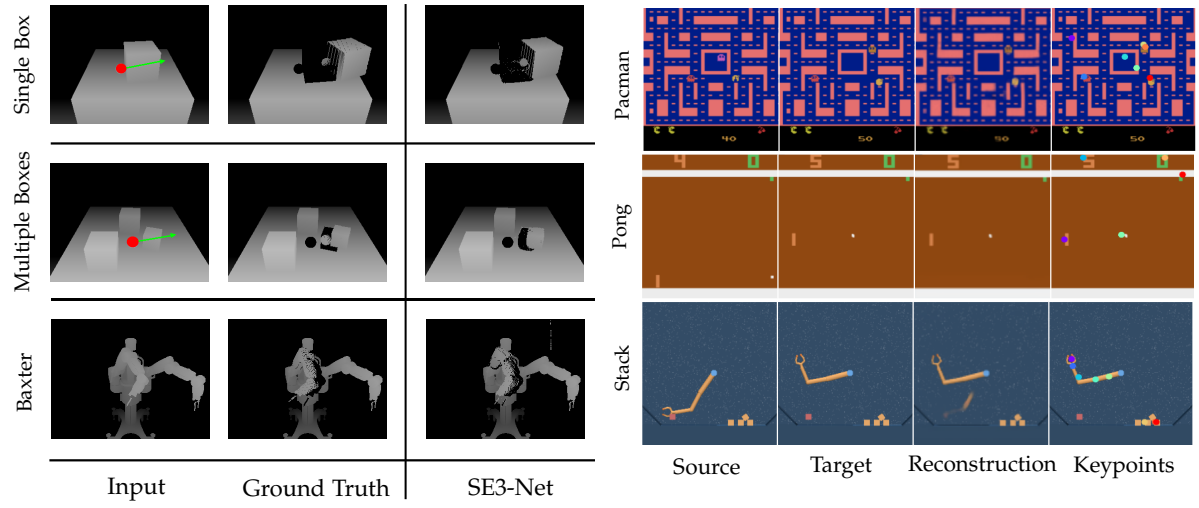
$$(i,j)[k+1] = \sum_{q=1}^{Q} (M^q)_{i,j}[k] \times \left(R^q(i,j)[k] + \boldsymbol{p}^q[k]\right) \tag{3-22}$$

The model is trained end-to-end to forecast future frames, see figure 3.7a for an illustration.

Using object-level transformations constrains the model to consider movements as a whole rather than pixel by pixel, which improves the performances in various applications, such as robotics manipulation, mapping (SLAM), and flow estimation (Vijayanarasimhan et al., 2017; Gojcic et al., 2021; Zhang et al., 2020a; Huang et al., 2021; Ryu et al., 2022). In a more general sense, one of the key interests of object-level representations is that they reduce dimensionality by transforming high-resolution images into meaningful low-dimensional vectors. These representations are arguably more suitable for intuitive physics, and can also offer some limited form of interpretability.

Similar to region-based approaches, the use of *keypoints* is a natural choice and is a popular object-level representation to describe movement with high precision. Historically, keypoints were computed using hand-crafted features extractor (Lowe, 1999), but can nowadays be discovered by deep learning. The main difficulty when detecting keypoints is that the dataset is usually not labeled, i.e. we do not have access to ground truth optimal keypoints within

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 3.7: **Object-level visual reasonning** – (a) **SE3-Net**: the input depth map and ground truth action (red dot and green arrow) are displayed in the first columns, the model outputs are shown in the last column. The model maintains object coherence by constraining all points belonging to the same motion mask to undergo the same transformation (figure from Byravan and Fox (2017)). (b) **Transporter Network**: keypoints (4th column) and reconstructed target frame (3rd column) obtained in three different setups. The discovered keypoints tracks the important part of the frames, such as the joints of the robot arm or the moving objects in the scene.

each frame to perform the downstream task, whatever it is. Note that the notion of *optimal keypoints* varies depending on the downstream task.

Transporter Network (Kulkarni et al., 2019) is an effective structure for building a keypoint detector suitable for control applications. In this model, keypoints are discovered through a reconstruction proxy task, which aims at reconstructing a target frame I_{target} from a source frame I_{source} using (1) **a feature extractor** $g_\theta(I)$ computing features from images using a CNN, and (2) **a keypoint detector** $k_\theta(I)$ which outputs 2D locations extracted from heatmaps, obtained by another CNN.

The model must discover points on relevant moving content, thus disentangling background (encoded into features) and object/point positions. This is achieved through a transport equation used to reconstruct the target image from the source, replacing the source features at keypoints locations with the ones from the target features:

$$z = \big(1 - k_\theta(I_{\text{target}})\big)\big(1 - k_\theta(I_{\text{source}})\big) \times g_\theta(I_{\text{source}}) + k_\theta(I_{\text{target}}) \times g_\theta(I_{\text{target}}) \tag{3-23}$$

The transported features z are used to reconstruct the target image using a de-convolution network. The model is trained end-to-end, minimizing the reconstruction loss, while inherently discovering meaningful 2D locations within the images. The Transporter network has been applied successfully for control and reinforcement learning tasks, including Atari

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

games (see figure 3.7b).

Keypoints have found practical applications in robotics manipulation, providing a convenient way to model joints and formulate high-level goals, with many applications in robotics (Zeng et al., 2021; Manuelli et al., 2019; Nagabandi et al., 2020; Manuelli et al., 2021). Moreover, keypoints are also employed for understanding videos of physical phenomena, particularly in rigid body dynamics to extract physical properties (e.g., mass, friction) and predict object movements (Minderer et al., 2019; Li et al., 2020a; Ehsani et al., 2020)

In chapter 9 of the manuscript, we extend the Transporter network by introducing shape coefficients and improving its temporal consistency. We demonstrate how causal reasoning can be learned using counterfactual learning, opening new possibilities for understanding complex dynamical systems from video data.

3.4 Learning the solver for mesh-based simulations

So far, we addressed simulations on regular spatial grids. We have discussed the limitations of such sampling strategy for physical observations, and shown different techniques to leverage object-level representations rather than pixel-based representations when dealing with videos. Yet, several problems from the real-world extends beyond visual reasoning.

Fortunately, many physical phenomena can be represented as a *connected graph of pointwise objects*, where each spatial node characterizes relevant physics properties. For instance:

- **Rigid body dynamics** are often simulated using 3D objects represented by connected polygons, allowing collision detection between objects.
- **Soft Body** simulations involve deformable objects, such as fluids, modeled as connected particles influenced by attractive or repulsive forces.
- **Fluid Dynamics & Electromagnetism** and other PDEs using eulerian formalism typically discretize space in a mesh of collocation points where physics quantities are measured.

The latter overcomes limitations faced by CNN-based methods. Convolution layers require a uniform discretization grid which us not suitable for adaptive resolution and cannot handle complex geometries. On the other hand, graph-based spatial discretization offers more versatility, as measurement points can be chosen arbitrarily to accommodate varying resolutions and intricate shapes.

Formally, let us introduce a temporal graph structure (with discrete time index k) modeling a physical problem $\mathcal{G}[k] = \{\mathcal{X}, \mathcal{A}, \mathcal{S}[k], \mathcal{E}[k]\}$, where $\mathcal{X} = \{x_1, x_2, \cdots\}$ are the vertex locations and $\mathcal{S}[k] = \{s_1[k], s_2[k], \cdots\}$ are the corresponding time-varying states. The graph also includes edges $\mathcal{A} = \{(i,j), \cdots\}$ that may be associated with features $\mathcal{E}[k] = \{e_{ij}[k], \cdots\}$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

to represent interaction between connected vertices (e.g. elasticity coefficient between two particles). Without loss of generality, we consider the case of a static graph structure (i.e. $\mathcal{X}$ and $\mathcal{A}$ do not depend on time).

There exists an entire field of deep learning dedicated to processing such graph-structured data called *geometric deep learning*. Geometric deep learning extends traditional neural networks with a *message passing* scheme which, in its simplest form, can be summarized in two equations[2], $\forall (i,j) \in \mathcal{A}$

$$\begin{array}{lll} \underline{\text{Message passing:}} & \boldsymbol{e}'_{ij} & = f^{\text{edge}}(\boldsymbol{s}_i, \boldsymbol{s}_j, \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{e}_{ij}) \\ \underline{\text{Update:}} & \boldsymbol{s}'_i & = f^{\text{node}}\left(\boldsymbol{s}_i, \boldsymbol{x}_i, \text{Agg}\left(\boldsymbol{e}'_{ij}\right)\right) \end{array}, \tag{3-24}$$

where f^{edge} and f^{node} are two neural networks, typically MLPs. The aggregation function $\text{Agg}\left(\boldsymbol{e}'_{ij}\right)$ merges the messages from the nodes j connected to node i (typically, a sum or an average operator). The design of the message passing step, the aggregation function, and the update step are not tailored to a specific design and can be adapted to the specificity of the task. We refer to this class of models as Graph Neural Network (GNN).

GNNs are widely used across various domains in deep learning, including physics (Shlomi et al., 2020; Satorras et al., 2021; Wang et al., 2022c), navigation (Lu et al., 2021; Chen et al., 2019; Beeching et al., 2020), and finance (Cheng et al., 2022; Matsunaga et al., 2019). The literature offers a diverse range of variants, leveraging spectral methods (Stachenfeld et al., 2020; Bianchi et al., 2020; Cao et al., 2020), attention mechanisms (Veličković et al., 2018; Wang et al., 2019c,d) or hierarchical structures (Zhang et al., 2020c,b).

For instance, to tackle rigid-body simulation, Allen et al. (2022b) extend the message-passing equations to consider not only node-to-node interactions but also edges-to-edges and faces-to-faces interactions. Unlike soft body dynamics, rigid bodies require fewer nodes for representing simple geometries (e.g., a cube needs only 8 points regardless of its size). By modeling faces-to-faces interactions, the model can accurately capture collisions that would have been challenging to detect solely from the node-level interactions.

However, there is a particular structure of GNN that seems to provide excellent simulation performance on a wide range of applications, the so-called *Encode-Process-Decode* pipeline. This structure processes graphs using three modules:

- The **encoder** projects the physical states in $\mathcal{S}$ in a higher-dimensional space. Formally, it computes embeddings $\boldsymbol{\eta}_i$ for each node. Edge features, if available, can also be embedded in another latent space, otherwise, they can be computed from scratch using geometrical consideration. The most common design is

$$\begin{array}{ll} \boldsymbol{\eta}_i & = g_\theta^{\text{node}}(\boldsymbol{x}_i, \boldsymbol{s}_i) \\ \boldsymbol{e}_{ij} & = g_\theta^{\text{edge}}(\boldsymbol{x}_i - \boldsymbol{x}_j, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \end{array}, \tag{3-25}$$

[2]Time dependency is omitted for readability

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

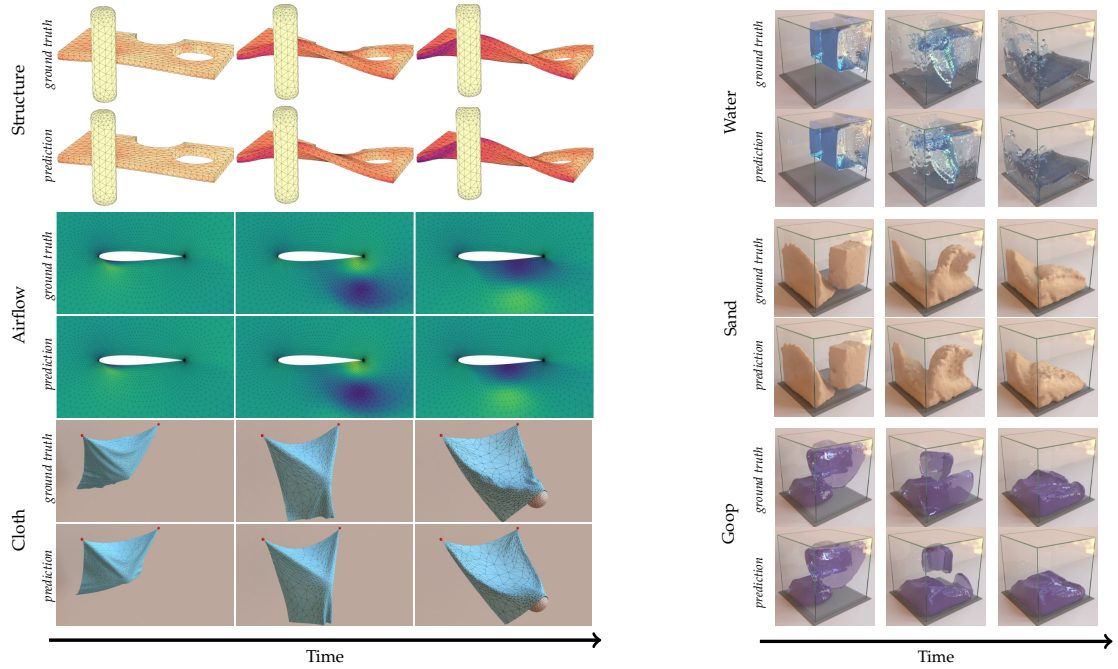


Figure 3.8: **Encode-Process-Decode** – Simulation results from (a) MeshGraphNet (Pfaff et al., 2020) and (b) Graph Network-based Simulator (GNS) (Sanchez-Gonzalez et al., 2020). MeshGraphNet simulates physics on meshes, which is a sensible way to model fluid dynamics, structure deformation, and clothes. GNS focuses on soft-body and smoothed-particle hydrodynamics and can simulate the behavior of fluid by tracking the evolution of a set of particles.

with g_θ^{node} and g_θ^{edge} are MLPs. The edges are usually computed using Delaunay triangulation or obtained from nearest neighbors.

- The **process** step consists in applying several layers of message-passing to the encoded graph (Scarselli et al., 2008; Kipf and Welling, 2016; Battaglia et al., 2016), i.e. multiple iterations of

$$\boldsymbol{e}_{ij} \leftarrow \boldsymbol{e}_{ij} + \overbrace{f_\theta^{\text{edge},\ell}(\boldsymbol{\eta}_i, \boldsymbol{\eta}_j, \boldsymbol{e}_{ij})}^{\boldsymbol{\varepsilon}_{ij}}, \tag{3-26}$$

$$\boldsymbol{\eta}_i \leftarrow \boldsymbol{\eta}_i + f_\theta^{\text{node},\ell}\left(\boldsymbol{\eta}_i, \sum_j \boldsymbol{\varepsilon}_{ij}\right), \tag{3-27}$$

where $f_\theta^{\text{edge},\ell}$ and $f_\theta^{\text{node},\ell}$ are two neural networks (MLPs) of the ℓth iteration.

- Finally, the **decoder** re-projects the resulting node embeddings $\boldsymbol{\eta}_i$ to the physical space. For simulation purposes, the output is usually embedded in a simple Euler scheme to advance the states forward in time

$$\boldsymbol{s}_i[k+1] = \boldsymbol{s}_i[k] + D_\theta(\boldsymbol{\eta}_i), \tag{3-28}$$

where D_θ is another MLP.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

This model has been successfully applied to soft body dynamics (Sanchez-Gonzalez et al. (2020), Graph Network-based Simulator (GNS)), and to mesh-based simulation, such as fluid dynamics, cloth simulation, and material deformation (Pfaff et al. (2020), MeshGraphNet). Both methods demonstrate impressive results (see figure 3.8). Li et al. (2018) propose a multi-scale graph network for simulating soft-body dynamics, while Han et al. (2021) incorporate temporal attention mechanisms to enhance the robustness of MeshGraphNet over longer horizons, albeit with reduced accuracy during initial steps of the rollout.

A promising application of such models lies in their differentiability property, allowing direct optimization of shapes to meet specific objectives using gradient descent on the geometry. This idea is explored in Allen et al. (2022a).

In our work, we contribute to the domain by introducing a mesh transformer applied to fluid simulations in chapter 8. Our approach specifically addresses a drawback of GNN-based models, which require many layers to propagate information over long distances.

3.5 Large scale datasets for physics

In the previous sections, we have presented physics and control-related models based on deep learning, but overlooked a crucial aspect, that is the necessity of large-scale datasets for training neural networks. Deep learning is known to be extremely data-intensive, which is a huge issue in the context of dynamical systems where generating data can be challenging.

In some cases, typically for control applications, it is possible to build a dataset of trajectories by numerically solving the ground truth ODE. In general, one might use a conventional solver (e.g. Runge-Kutta methods) to generate a large number of trajectories from random initial conditions (Gilpin, 2021; Gaby et al., 2022; Peralez et al., 2022). To ensure precision, a good practice consists of (1) simulating the trajectory with a smaller time step than the desired one (by roughly an order of magnitude), and (2) down-sampling the result to the targeted rate.

Challenges arise when dealing with dynamics in the form of a PDE, as conventional numerical solvers become time-consuming, which might limit the precision or the number of simulated trajectories in the dataset. Fluid mechanics is a good example: simulation data can be acquired through several solvers, ranging from computer graphics-oriented simulators (Takahashi et al., 2021; Pfaff and Thuerey, 2016) to accurate computational fluid dynamics solver (OpenFOAM©, Ansys© Fluent, ...). A large body of work (Chen et al., 2021a; Pfaff et al., 2020; Han et al., 2021; Stachenfeld et al., 2021) introduces synthetic datasets, yet is limited to simple tasks such as 2D flow past a cylinder. Accurate 3D simulations are mostly conducted on grid-based meshes and for rather simple, theoretic problems (Mohan et al., 2020; Chen et al., 2021b; Stachenfeld et al., 2021). The John Hopkins Turbulent Database (Li et al., 2008) contains nine direct numerical simulation datasets (i.e. direct resolution of Navier-Stokes equations) but with only a single scene per dataset simulated on a very fine grid at low time

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

resolution.

In rigid body mechanics, numerical simulation involves collisions and resting contacts of moving objects which are not trivial to handle. These datasets can serve various purposes and are not limited to the collection of trajectories: Kubric (Greff et al., 2022) is a dataset generator for rigid body scenes which leverages PyBullet© for physics simulation and Blender© for visual rendering. CLEVRER (Yi et al., 2019) is a visual question-answering dataset, where an agent is required to answer a counterfactual question after observing a video showing 3D objects moving and colliding and PHYRE (Bakhtin et al., 2019) is a physics benchmark involving agent interaction to achieve collision-related goals.

In this thesis, we present two challenging datasets. In chapter 8, we introduce a fluid mechanics dataset that has required several months of simulations on a high-end computer to simulate a large-scale dataset of a difficult prediction task. In chapter 9, we propose a counterfactual physics dataset based on rigid body mechanics, derived from CoPhy (Baradel et al., 2020). The difficulty of generating this dataset lies in the assessment of the feasibility of the underlying task.

Obtaining datasets from real physical systems is rare and difficult. Some methods leverage alignment with numerical simulations to extrapolate ground truth flows on real-world phenomena (Eckert et al., 2019; Bézenac et al., 2019). Lerer et al. (2016) harnessed a handful of real-world rigid body dynamics by filming towers of wooden cubes in a hundred different configurations. Several real-world datasets for mechanical and electronic devices are also available but are limited in size (Weigand et al., 2023; Janot et al., 2019; Schoukens and Noël, 2017). For drone control, highly accurate motion tracking devices have been used to create large-scale datasets of drone flights (Song et al., 2023; Cioffi et al., 2023; Pfeiffer et al., 2022; Loquercio et al., 2021; Bauersfeld et al., 2021).

3.6 Take-home messages

This chapter focuses on specialized models designed to address intuitive physics. We discussed two different approaches: learning the solution and learning the solver. For systems governed by known differential equations, PINNs look for a solution using neural networks in a mesh-free fashion. However, this method cannot generalize to new initial conditions. When the governing equation is unknown, input-dense solutions can still be achieved using discrete measurement datasets, but this requires high modeling efforts.

Deep learning excels in auto-regressive prediction tasks and extends its capabilities to image-based reasoning and fluid mechanics. CNNs are particularly suited for video processing, which is challenging to address with conventional tools. For better performance, training CNNs can be biased toward object-centric representations, such as keypoints.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

In a more general setup, the advantages of object-level representations suggest using geometric deep learning and graph neural networks to simulate physics. Such models achieve impressive results on complex phenomena, such as rigid or soft body mechanics. However, training these large-scale models necessitates large-scale datasets, which can be particularly challenging to obtain in the context of physics-related tasks.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Part II

Foundations for Robust Simulations using Observer Theory

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

General remarks

The identification of nonlinear dynamical models is an open topic in control theory, especially from sparse input-output measurements. A fundamental challenge of this task is that, in the general case, very little to zero prior knowledge is available on both the state and the nonlinear system. Moreover, many applications require specific guarantees on the identification method, such as the existence and uniqueness of the solution or error bounds. In what follows, we introduce new theoretical tools and constructive methods for the identification of nonlinear dynamical systems with neural networks. In particular, we rely on observer theory to build models that maintain certain essential guarantees. In practice, our objective is to perform open-loop simulation with high accuracy.

Chapter 4: Learning Reduced Nonlinear State-Space Models: an Output-Error Based Canonical Approach

TLDR; We show that, under some structural conditions on the to-be-identified model, the state can be expressed as a function of a sequence of past inputs and outputs. This relation, which we call the state map, can be modeled as a neural network. Taking advantage of existing learning schemes, a state-space model can be identified. After the formulation and analysis of the approach, we show its ability to identify three different nonlinear systems.

Chapter 5: Deep KKL: Data-driven Output Prediction for Non-Linear Systems

TLDR; We address the problem of output prediction and define a general framework bringing together the necessary properties for such an output predictor. We try to formulate it consistently, reducing the gap between deep learning and control theory. Building on this formulation and problem definition, we propose a predictor structure based on the Kazantzis-Kravaris/Luenberger (KKL) observer and we show that it fits well into our setup. Finally, we propose a constructive solution for this predictor that solely relies on a small set of trajectories measured from the system.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 4

Learning Reduced Nonlinear State-Space Models: an Output-Error Based Canonical Approach

Work presented at Conference on Decision and Control 2022,
Co-authors: Quentin Possamaï (Alstom),
Laurent Bako (Centrale Lyon),
Madiha Nadri (Université Lyon 1),
Christian Wolf (NaverLabs Europe)

4.1 Context

Most approaches in control theory commonly build upon a deterministic model that describes how the state variables evolve. This model holds a central role, as the effectiveness of the method frequently hinges on its accuracy (Weinmann, 2012; Cheah et al., 2006; Bauersfeld et al., 2021; Buşoniu et al., 2018; Bemporad, 2006). As a result, modeling and identifying a dynamical system is the cornerstone of downstream algorithms, such as controller or observer design. This is not a trivial task: physical systems are typically complex, often non-linear, and require a trade-off between thorough modeling of physical phenomena and computation time. On the other hand, the identification of the parameters of a non-linear model is a non-convex problem, which can require tremendous hours of calibrations and experiments. Moreover, conventional dynamical modeling often requires domain experts and the ability to freely interact with the system.

Data-driven techniques for the identification of non-linear systems show encouraging results against these challenges (Ljung et al., 2020; Masti and Bemporad, 2018; Pillonetto et al., 2014). Specifically, deep learning offers a change of point of view, redirecting painstaking efforts in physical modeling toward the collection of large-scale datasets of trajectories from the system. The main insight is to rely on extremely versatile parametric functions (i.e. neural networks in our case) capable of approaching most dynamics up to a certain degree of precision. The parameters can be directly identified from pairs of input-output measurements, provided that these measurements gather enough information to approximate the true dynamics. Nevertheless, the great flexibility of neural networks comes at the cost of a lack of mathematical

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

structure making it difficult to derive theoretical results in terms of robustness, precision, and stability. Moreover, learning complex, high-dimensional dynamical systems is not straightforward. The general formulation leads to latent dynamics models lacking meaningful physical structure and requires large dimensional state spaces.

In this chapter, we propose an identification structure for nonlinear state-space systems from a set of observation trajectories and associated inputs. We demonstrate the existence of a regressor inspired by finite impulse response models allowing us to map a series of past observations to future outputs and provide bounds derived from the prediction error during deployment. We then deduce a high-dimensional *canonical state-space model* discovered using an output-error-based approach and propose to learn an auto-encoder projecting the dynamics into a smaller state-space. We evaluate our proposal on different systems in simulation and the real world.

We now recall the main related works already reviewed with more details in chapter 2. Data-driven dynamics are widely studied in the community and get a lot of attention. In particular, Brunton et al. (2016); Sahoo et al. (2018); Chen et al. (2021d) propose to find governing equations by performing a sparse regression from the data. At the junction between physical modeling and learning, Yin et al. (2021b); Long et al. (2018); Wang et al. (2019b); Mehta et al. (2021b) use neural networks to model complementary phenomena not described by the initial physical model. For instance, Shi et al. (2019); Bauersfeld et al. (2021); Possamaï et al. (2022) extend the dynamical model of a Unmanned Aerial Vehicle (UAV) with a neural network in charge of predicting aerodynamic disturbances, which are often very demanding and intractable for real-time physical simulation, when addressed with conventional methods.

Close to our work, a body of literature proposes to use deep learning for the identification of latent dynamics, i.e. without direct physical meaning of the (latent) state variable. This is notably the case for recent work around the Koopman operator (Lusch et al., 2018; Janny et al., 2021; Peralez et al., 2020; Rowley et al., 2009). Another solution is to use an auto-encoder structure to model the latent dynamics of a system from past observations (Masti and Bemporad, 2018; Beintema et al., 2021). Our proposal differs from this line of work in three main points: (1) we provide theoretical results and conditions for the existence of the dynamical system that we identify, (2) we propose to use a high-dimensional regressor structure without explicit state representation, which will be deduced from a dimensionality reduction operation and (3) we evaluate our approach on challenging and unstable systems.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

4.2 Problem statement and preliminary results

4.2.1 *Problem statement*

We consider a nonlinear discrete-time system of the general form:

$$\begin{cases} \boldsymbol{s}[k+1] &= f\big(\boldsymbol{s}[k], \boldsymbol{u}[k]\big) \\ \boldsymbol{y}[k] &= h\big(\boldsymbol{s}[k], \boldsymbol{u}[k]\big) + \boldsymbol{w}[k] \end{cases}, \tag{4-1}$$

with $\boldsymbol{s}[k] \in \mathcal{S} \subset \mathbb{R}^{n_s}$, $\boldsymbol{u}[k] \in \mathcal{U} \subset \mathbb{R}^{n_u}$, and $\boldsymbol{y}[k] \in \mathcal{Y} \subset \mathbb{R}^{n_y}$ being the state, the input and the output of the system at discrete time $k \in \mathbb{N}$ respectively. $f : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_s}$ and $h : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ are some nonlinear vector-valued functions. As to $\boldsymbol{w}[k] \in \mathcal{W} \subset \mathbb{R}^{n_y}$, it represents measurement noise. We make the following assumptions:

1. The external signals $\boldsymbol{u}$ and $\boldsymbol{w}$ take values in compact sets $\mathcal{U}$ and $\mathcal{W}$ respectively.
2. The state-space $\mathcal{S}$ is a known compact set containing the initial state $\boldsymbol{s}_0$.
3. $(\mathcal{S}, \mathcal{U}, \mathcal{W}, \mathcal{Y})$ and (f, h) satisfy the following invariance conditions:
$$\begin{aligned} &\forall (\boldsymbol{s}, \boldsymbol{u}) \in \mathcal{S} \times \mathcal{U}, && f(\boldsymbol{s}, \boldsymbol{u}) \in \mathcal{S} \\ &\forall (\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{w}) \in \mathcal{S} \times \mathcal{U} \times \mathcal{W}, && h(\boldsymbol{s}, \boldsymbol{u}) + \boldsymbol{w} \in \mathcal{Y} \end{aligned} \tag{4-2}$$
4. f and h are uniformly Lipschitz continuous on $\mathcal{S} \times \mathcal{U} \subset \mathbb{R}^{n_s} \times \mathbb{R}^{n_u}$ with respect to $\mathcal{U}$, i.e., there exists a constant $\gamma_f > 0$ such that $\|f(\boldsymbol{s}, \boldsymbol{u}) - f(\boldsymbol{s}', \boldsymbol{u})\| \leqslant \gamma_f \|\boldsymbol{s} - \boldsymbol{s}'\|$ for all $(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{u}) \in \mathcal{S}^2 \times \mathcal{U}$.

The assumptions 1 to 4 are essentially required to theoretically ensure the well-definedness of optimization problems that will be expressed later in the chapter. Assumptions 1 to 3 guarantee that the physical state remains in a compact set, which is a reasonable assumption for physical systems. Assumption 4 concerns the smoothness of the dynamics, and excludes too abrupt changes, such as a ball bouncing against a wall. Of course in the context of system identification, such assumptions are not intended to be checked before applying the method to be developed.

The problem of interest in this chapter can be stated as:

Given a finite number N of input-output data pairs $\Big\{ \big(\boldsymbol{u}[k], \boldsymbol{y}[k]\big) \Big| k \in [\![1, K]\!] \Big\}$ generated by a nonlinear system of the form (4-1), and under assumptions 1 to 4, find an appropriate dimension n_s of a state-space representation along with estimates of the associated functions f and h.

Here, the dimensions n_y of outputs and n_u of inputs are known a priori. However, the dimension n_s of the state is a parameter of the model that needs to be estimated along with the maps (f, h). We develop a solution in three steps: first, a nonlinear regression model is derived from the system equations (4-1). The underlying non-linear map is then modeled by a deep neural network and trained with the available data following an output-error principle.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Given this map, we derive an equivalent canonical state-space representation of system (4-1) typically of high dimension. Hence, the third and last step of the proposed procedure consists of model reduction, i.e. the reduction of the state dimension. The process aims at finding another state-space model that is as close as possible to the primary one but with a lower dimension. This is achieved through the design of an appropriate encoder-decoder.

4.2.2 *Preliminary results*

An important challenge concerning the identification of the system (4-1) is the fact that the state $\boldsymbol{s}[k]$ is not entirely measured. We therefore need to express it first as a function of the available past input-output measurements $\left\{(\boldsymbol{u}[\tau],\boldsymbol{y}[\tau])\middle|\tau<k\right\}$. Indeed, if the noise $\boldsymbol{w}[k]$ in system (4-1) is assumed to be identically equal to zero, then under appropriate observability conditions on the system, there exists a time horizon ℓ and a map $\phi:\mathbb{R}^L\to\mathbb{R}^{n_s}$, with $L=\ell(n_u+n_y)$, such that the state $\boldsymbol{s}[k]$ can be written as

$$\boldsymbol{s}[k]=\phi(\boldsymbol{z}[k])\quad\text{where}\quad \boldsymbol{z}[k]=\begin{bmatrix}\boldsymbol{u}[k-\ell]^\top & \boldsymbol{y}[k-\ell]^\top & \cdots & \boldsymbol{u}[k-1]^\top & \boldsymbol{y}[k-1]^\top\end{bmatrix}^\top \tag{4-3}$$

is the so-called regressor vector. To show the existence of such a map ϕ, some observability conditions on the system to be identified are needed. For this purpose let us start by introducing some preliminaries. For a positive integer k, let $F_k:\mathbb{R}^{n_s}\times\mathbb{R}^{k\times n_u}\to\mathbb{R}^{n_s}$ be the map defined recursively from the function f in equation (4-1) as follows: for $\boldsymbol{s}\in\mathbb{R}^{n_s}$ and $(\boldsymbol{u}[1],\dots,\boldsymbol{u}[k])\in\mathbb{R}^{k\times n_u}$, $F_1(\boldsymbol{s},\boldsymbol{u}[1])=f(\boldsymbol{s},\boldsymbol{u}[1])$ and for all $k\geqslant 2$,

$$F_k(\boldsymbol{s},\boldsymbol{u}[1],\dots,\boldsymbol{u}[k])=f\Big(F_{k-1}(\boldsymbol{s},\boldsymbol{u}[1],\dots,\boldsymbol{u}[k-1]),\boldsymbol{u}[k]\Big). \tag{4-4}$$

Before proceeding further, let us mention a useful property of the maps F_k.

Lemma 4.1 *Under Assumption 3, if $f:\mathbb{R}^{n_s}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_s}$ is uniformly γ_f-Lipschitz on $\mathcal{S}\times\mathcal{U}$ with respect to $\mathcal{U}$, then the map F_k defined in equation (4-4) is uniformly γ_f^k-Lipschitz on $\mathcal{S}\times\mathcal{U}^k$ with respect to $\mathcal{U}^k\subset\mathbb{R}^{k\times n_u}$.*

Proof: The proof of this lemma is straightforward and is therefore omitted. ■

Now consider the function $\mathcal{O}_k:\mathbb{R}^{n_s}\times\mathbb{R}^{k\times n_u}\to\mathbb{R}^{k\times n_y}$ given by

$$\mathcal{O}_k(\boldsymbol{s},\boldsymbol{u}[1],\dots,\boldsymbol{u}[k])=\begin{bmatrix}h(\boldsymbol{s},\boldsymbol{u}[1])\\ h\Big(F_1(\boldsymbol{s},\boldsymbol{u}[1]),\boldsymbol{u}[2]\Big)\\ \vdots\\ h\Big(F_{k-1}(\boldsymbol{s},\boldsymbol{u}[1],\dots,\boldsymbol{u}[k-1]),\boldsymbol{u}[k]\Big)\end{bmatrix}. \tag{4-5}$$

For notational simplicity, let us define the stacked vector $\bar{\boldsymbol{u}}[1|k]=\begin{bmatrix}\boldsymbol{u}[1]^\top & \dots & \boldsymbol{u}[k]^\top\end{bmatrix}^\top$ so that $\mathcal{O}_k(\boldsymbol{s},\boldsymbol{u}[1],\dots,\boldsymbol{u}[k])$ in the previous equality can be replaced by $\mathcal{O}_k(\boldsymbol{s},\bar{\boldsymbol{u}}[1|k])$.

Definition 4.1 *The system (4-1) is said to be finite-time observable over a time horizon $r\in\mathbb{N}$ if and only if for each $\bar{\boldsymbol{u}}\in\mathcal{U}^r$, the function $\mathcal{O}_r(\cdot,\bar{\boldsymbol{u}})$, with $\mathcal{O}_r$ defined as in equation (4-5), is injective.*

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Note that if the observability property in Definition 4.1 holds for some $r \in \mathbb{N}$ then it holds as well for any $k \geqslant r$.

Proposition 4.1 (Existence of the map ϕ) *If the nonlinear system (4-1) (considered under the assumption that $\boldsymbol{w} \equiv 0$) is finite-time observable in the sense of Definition 4.1, then there exist $\ell \in \mathbb{N}$ and a (non-linear) map $\phi : \mathbb{R}^L \to \mathbb{R}^{n_s}$ such that equation (4-3) holds for all time $k \geqslant \ell$, any initial state in $\mathcal{S}$ and any input signal taking values in $\mathcal{U}$.*

Proof: By iterating the system equations, it is easy to see that

$$\bar{\boldsymbol{y}}[k-\ell|k-1] = \mathcal{O}_\ell\big(\boldsymbol{s}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big). \tag{4-6}$$

By the finite-time observability assumption of the system, $\mathcal{O}_\ell(\cdot, \bar{\boldsymbol{u}})$ admits an inverse for any given $\bar{\boldsymbol{u}} \in \mathcal{U}^\ell$. Denote with $\mathcal{O}_\ell^*(\cdot, \bar{\boldsymbol{u}}) : \mathbb{R}^{\ell \times n_y} \to \mathbb{R}^{n_s}$ the inverse map of $\mathcal{O}_\ell(\cdot, \bar{\boldsymbol{u}})$ which is such that $\mathcal{O}_\ell^*\left(\mathcal{O}_\ell(\boldsymbol{s}, \bar{\boldsymbol{u}}), \bar{\boldsymbol{u}}\right) = \boldsymbol{s}$. It hence follows from equation (4-6) that

$$\boldsymbol{s}[k-\ell] = \mathcal{O}_\ell^*\big(\bar{\boldsymbol{y}}[k-\ell|k-1], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) \tag{4-7}$$

which, by recursively applying the first equation of (4-1), gives

$$\boldsymbol{s}[k] = F_\ell\Big(\mathcal{O}_\ell^*\big(\bar{\boldsymbol{y}}[k-\ell|k-1], \bar{\boldsymbol{u}}[k-\ell|k-1]\big), \bar{\boldsymbol{u}}[k-\ell|k-1]\Big) := \phi\big(\boldsymbol{z}[k]\big). \tag{4-8}$$

■

Consider now the more realistic scenario where the (unknown) measurement noise sequence $\{\boldsymbol{w}[k]\}$ is nonzero. Then equation (4-6) becomes

$$\bar{\boldsymbol{y}}[k-\ell|k-1] = \mathcal{O}_\ell\big(\boldsymbol{s}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) + \bar{\boldsymbol{w}}[k-\ell|k-1]. \tag{4-9}$$

As a consequence, the state can no longer be obtained exactly by equation (4-7) or (4-8) since $\bar{\boldsymbol{y}}[k-\ell|k-1]$ does not lie in the range of $\mathcal{O}_\ell\big(\cdot, \bar{\boldsymbol{u}}[k-\ell|k-1]\big)$. Let in this case the state $\boldsymbol{s}[k-\ell]$ and $\boldsymbol{s}[k]$ be estimated by

$$\hat{\boldsymbol{s}}[k-\ell] \in \arg\min_{\boldsymbol{s}\in\mathcal{S}} \Big|\bar{\boldsymbol{y}}[k-\ell|k-1] - \mathcal{O}_\ell\big(\boldsymbol{s}, \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\Big| \tag{4-10}$$

$$\hat{\boldsymbol{s}}[k] = F_\ell\big(\hat{\boldsymbol{s}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big), \tag{4-11}$$

for some norm $|\cdot|$ on $\mathbb{R}^{\ell \times n_y}$. The optimization problem (4-10) is well-defined since, by Assumptions 1 to 4, the function $\boldsymbol{s} \mapsto \Big|\bar{\boldsymbol{y}}[k-\ell|k-1] - \mathcal{O}_\ell\big(\boldsymbol{s}, \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\Big|$ is defined on a compact set $\mathcal{S}$ and is continuous. Using the extreme value theorem, it gives sufficient conditions for the existence of a minimum and for the existence of the minimizer $\hat{\boldsymbol{s}}[k-\ell]$ as defined above.

In contrast, the estimates $\hat{\boldsymbol{s}}[k-\ell]$ and $\hat{\boldsymbol{s}}[k]$ may not be uniquely defined in a general setting. Uniqueness would require some more strict conditions on the system. Here, we will be content with a set-valued version $\hat{\phi}$ of ϕ in the noisy estimation scenario. Hence let $\hat{\phi}$ be defined by

$$\hat{\phi}\big(\boldsymbol{z}[k]\big) = \Big\{F_\ell\big(\hat{\boldsymbol{s}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\Big|\hat{\boldsymbol{s}}[k-\ell] \text{ as in equation (4-10)}\Big\}. \tag{4-12}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

A question we ask now is how far the noisy estimate from equation (4-11) lies from the true state $s[k]$. To study this, a stronger notion of observability is introduced as follows.

Definition 4.2 *The system (4-1) is called finite-time uniformly observable over a time horizon $\ell \in \mathbb{N}$ if there exists a constant $\alpha_\ell > 0$ such that for each $\bar{u} \in \mathcal{U}^\ell$,*

$$\|\mathcal{O}_\ell(s, \bar{u}) - \mathcal{O}_\ell(s', \bar{u})\| \geqslant \alpha_\ell \|s - s'\| \tag{4-13}$$

for all $(s, s') \in \mathbb{R}^{n_s} \times \mathbb{R}^{n_s}$. Here $\|\cdot\|$ denotes a generic norm defined on appropriate spaces.

Informally, this definition measures how difficult it is to distinguish two initial conditions from their respective observed outcomes. The value of α_ℓ can be seen as a measurement of the injectivity of $\mathcal{O}_\ell$, i.e. α_ℓ close to zero implies $\mathcal{O}_\ell$ weakly injective. Based on this property, it is possible to bound the error between the noisy estimate (4-11) and the true state.

Proposition 4.2 *Under Assumptions 1 to 4, if the system (4-1) is finite-time uniformly observable over a time horizon $\ell \in \mathbb{N}$ in the sense of Definition 4.2, then*

$$\left|\hat{s}[k] - s[k]\right| \leqslant 2\gamma_f^\ell \alpha_\ell^{-1} \left|\bar{w}[k-\ell|k-1]\right| \tag{4-14}$$

where γ_f is the Lipschitz constant of f (See Assumption 4) and α_ℓ is the constant appearing in equation (4-13).

Proof: See appendix A.1. ■

The result follows now by applying equation (4-11), the uniform Lipschitz assumption on f stated in assumption 4 and Lemma 4.1. It can be seen from the expression of the error bound (4-14) that the more observable the system is (that is, the larger the constant α_ℓ), the more robust the estimate $\hat{s}[k]$. Indeed α_ℓ, when it exists, can be defined as

$$\inf_{\substack{\bar{u}, s, s' \in \mathcal{U}^\ell \times \mathcal{S} \times \mathcal{S} \\ s \neq s'}} \frac{\left|\mathcal{O}_\ell(s, \bar{u}) - \mathcal{O}_\ell(s', \bar{u})\right|}{\left|s - s'\right|}. \tag{4-15}$$

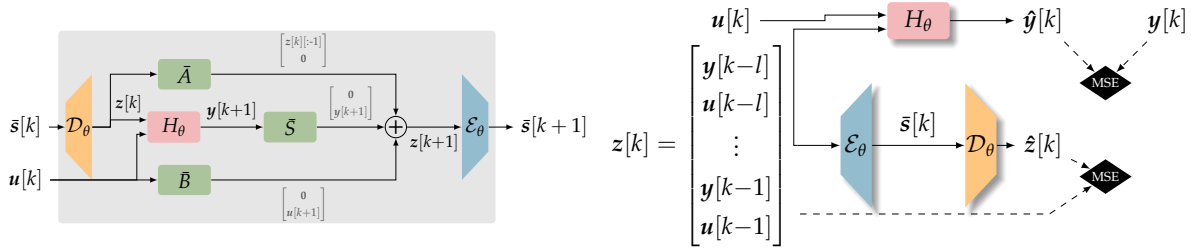
4.3 Modeling and learning

4.3.1 *Nonlinear regression model*

A starting point of our identification method for system (4-1) is to solve a nonlinear regression problem. To formulate this, note by Proposition 4.2 that the true state of the system can be written as $s[k] = \hat{s}[k] + \delta[k]$ with $\left|\delta[k]\right| \leqslant \gamma_f^\ell \alpha_\ell^{-1} \left|\bar{w}[k-\ell|k-1]\right|$. Consider now plugging the state estimate (4-11) into the output equation of system (4-1), which gives

$$\begin{aligned} y[k] &= h\big(\hat{s}[k] + \delta[k], u[k]\big) + w[k] \\ &= h\big(\hat{s}[k], u[k]\big) + \xi[k], \end{aligned} \tag{4-16}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 4.1: **Model overview and training pipeline** – (a) Block diagram of our canonical reduced state space representation as defined in equation (4-22) (b) Training: the dynamics is modeled by H_θ acting as a regressor from a short history of previous observations to the future value. The encoder-decoder model is used to reduce the size of the state $z[k]$ to $s[k]$. We train each network by minimizing the prediction error as well as the reconstruction error.

with $\boldsymbol{\xi}[k]$ being an error component entirely due to the noise. It is indeed equal to zero whenever $\boldsymbol{w} \equiv 0$. It can be shown that $\boldsymbol{\xi}[k]$ can be written as $\boldsymbol{\xi}[k] = \boldsymbol{w}[k] + \tilde{\delta}[k]$ with $\left|\tilde{\delta}[k]\right| \leqslant \gamma_h \gamma_f^\ell \alpha_\ell^{-1} \left|\bar{\boldsymbol{w}}[k-\ell|k-1]\right|$, where γ_h is the Lipschitz constant of the measurement function h of system (4-1). Since $\hat{\boldsymbol{s}}[k]$ is a function of $\boldsymbol{z}[k]$ we end up with

$$\boldsymbol{y}[k] = H\big(\boldsymbol{z}[k], \boldsymbol{u}[k]\big) + \boldsymbol{\xi}[k] \tag{4-17}$$

for some nonlinear function H, referred to as the *regressor function*.

In the absence of noise, the exact expression of H is

$$H\big(\boldsymbol{z}[k], \boldsymbol{u}[k]\big) = h\Big(F_\ell\Big(\mathcal{O}_\ell^*\big(\boldsymbol{z}[k]\big), \eta\big(\boldsymbol{z}[k]\big)\Big), \boldsymbol{u}[k]\Big), \tag{4-18}$$

with $\eta\big(\boldsymbol{z}[k]\big) = \bar{\boldsymbol{u}}[k-\ell|k-1]$.

The second step of the identification method is to construct a high dimensional state-space representation whose state is the vector $\boldsymbol{z}[k]$ defined in equation (4-3). More precisely, consider

$$\begin{cases} \boldsymbol{z}[k+1] &= \bar{A}\boldsymbol{z}[k] + \bar{B}\boldsymbol{u}[k] + \bar{S}H\big(\boldsymbol{z}[k], \boldsymbol{u}[k]\big) + \bar{S}\boldsymbol{\xi}[k] \\ \boldsymbol{y}[k] &= H\big(\boldsymbol{z}[k], \boldsymbol{u}[k]\big) + \boldsymbol{\xi}[k], \end{cases} \tag{4-19}$$

where $\bar{A} = A \otimes I_{n_u+n_y}$, $\bar{B} = \boldsymbol{e}_{n_s-1} \otimes I_{n_u}$, $\bar{S} = \boldsymbol{e}_{n_s} \otimes I_{n_y}$, $\boldsymbol{e}_i \in \mathbb{R}^{n_s}$ being the canonical basis vector which has 1 in its i-th entry and zero everywhere else, $\otimes$ referring to the Kronecker product and $A \in \mathbb{R}^{n_s \times n_s}$ given by the canonical form

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \ddots & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix}. \tag{4-20}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

The matrices $\bar{A}, \bar{B}$, and $\bar{S}$ are simply used to crop and shift the vector $z[k]$. Namely, $\bar{A}$ removes the last input-output pair and shifts the remaining ones to the bottom of the vector. $\bar{B}$ shifts the new input $u[k]$ in place of the last input control, and $\bar{S}$ does the same on the predicted output $H\big(z[k], u[k]\big)$. By summing each component, we obtain the next vector $z[k+1]$.

From equation (4-17) it can be seen that the system (4-19) constitutes a state-space representation for system (4-1) since both models have the same input-output behavior for $k \geqslant \ell$. Given a finite set of input-output data points $\Big\{ \big(u[k], y[k]\big) \Big| k \in [\![1, K+\ell]\!] \Big\}$ for some finite observation window $[\![0, K+\ell]\!]$, an estimate H_θ of the function H can be obtained in a certain nonlinear model class $\mathcal{H}$ by minimizing a regression loss $J(H_\theta)$ given as

$$J(H_\theta) = \frac{1}{K} \sum_{k=\ell+1}^{K+\ell} \beta[k] \Big| y[k] - H_\theta\big(\hat{z}[k], u[k]\big) \Big| \text{ s.t. } \begin{cases} \hat{z}[k+1] &= \bar{A}\hat{z}[k] + \bar{B}u[k] + \bar{S}H_\theta\big(\hat{z}[k], u[k]\big), \\ \hat{z}[\ell+1] &= z[\ell+1] \end{cases} \tag{4-21}$$

where $\beta[k]$ is a weighting coefficient such as $\beta[k]=1$ except for $\beta[\ell+1] = 10$. Regression starts after a burn-in phase of ℓ steps (i.e. the window size), which are needed to construct a full state representation.

4.3.2 *Model reduction*

The last step of the proposed method consists of model reduction. Indeed, the model described in equation (4-19), although structurally simple, may suffer from a high dimensional state vector $z[k]$. This may be a concern for some applications. We therefore propose a second deep learning structure allowing non-linear state-space model reduction by encoding the state variable $z[k]$ into a low dimensional state variable $\bar{s}[k] \in \mathbb{R}^{\bar{n}}$ for some user-defined dimension $\bar{n} \in \mathbb{N}$. Formally, we train an auto-encoder $(\mathcal{E}_\theta, \mathcal{D}_\theta)$ such that $\bar{s}[k] = \mathcal{E}_\theta\big(z[k]\big)$ and $z[k] = \mathcal{D}_\theta\big(\bar{s}[k]\big)$. By applying these maps to equation (4-19) and neglecting the noise terms, we get an approximate representation of the initial system (4-1) as follows:

$$\begin{cases} \bar{s}[k+1] &= \mathcal{E}_\theta\Big(\bar{A}\mathcal{D}_\theta\big(\bar{s}[k]\big) + \bar{B}u[k] + \bar{S}H_\theta\Big(\mathcal{D}_\theta\big(\bar{s}[k]\big), u[k]\Big)\Big) \\ \bar{y}[k] &= H_\theta\Big(\mathcal{D}_\theta\big(\bar{s}[k]\big), u[k]\Big). \end{cases} \tag{4-22}$$

The state-space equation is summarized in figure 4.1a. The parameters of the encoder $\mathcal{E}_\theta$ and decoder $\mathcal{D}_\theta$ are trained with the reconstruction loss from data samples $\{z[k]\}$ collected from the training set (see also Fig. 4.1b, i.e. $\Big| z[k] - \mathcal{D}_\theta\Big(\mathcal{E}_\theta\big(z[k]\big)\Big) \Big|$.

Another observation is that by going from equation (4-19) to equation (4-22), one reduces the dimension of the state vector but at the cost of introducing some structural complexity. Hence the computational cost associated with simulating a model such as system (4-22) may still be high depending on the complexity of the auto-encoder $(\mathcal{E}_\theta, \mathcal{D}_\theta)$.

From a formal point of view, this reduced state has several advantages. The constructed

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

state-space $z[k]$ is not part of the update equation (4-22) anymore. We can also experimentally show (see section 4.4), that this method can discover state representations of smaller size with a method that is generic in nature and can be applied to a broad class of problems.

4.3.3 *Model architecture*

The choice of the model class $\mathcal{H}$ is fundamental for several reasons: it must be sufficiently large to represent a good approximation of H, and it should not be too large to ensure learnability and generalization to unseen conditions (Shalev-Shwartz and Ben-David, 2014). In other words, this class needs to be complex enough to capture the behavior of the unknown nonlinear function H while being easily identifiable from a rather limited set of measurements.

We thus propose two different implementations of the regressor function H_θ. First, we rely on the well-documented approximation power of MLPs, whose capacity shall be limited. In particular, the number of hidden layers and number of neurons of each layer is considered as a hyper-parameter optimized over a validation set independent of training and evaluation splits, as classically done in machine learning. We refer to this design as *Ours (MLP)*.

$$\text{Ours (MLP):} \quad \begin{cases} \boldsymbol{z}[k+1] &= \bar{A}\boldsymbol{z}[k] + \bar{B}\boldsymbol{u}[k] + \bar{S}g_\theta\big(\boldsymbol{z}[k], \boldsymbol{u}[k]\big), \\ \boldsymbol{y}[k] &= g_\theta\big(\boldsymbol{z}[k], \boldsymbol{u}[k]\big), \end{cases} \tag{4-23}$$

with g_θ being an MLP. We also introduce an extension of our model using our proposed state representation $z[k]$, but implement the mapping H by a GRU. Formally, the GRU updates a zero-initialized hidden vector using the previous observations and control input. This vector is then decoded by a MLP to the desired observation. Equation (4-19) is then used for forward prediction. We refer to this model as *Ours (GRU)*, it is given as

$$\text{Ours (GRU):} \quad \begin{cases} \boldsymbol{z}[k+1] &= \bar{A}\boldsymbol{z}[k] + \bar{B}\boldsymbol{u}[k] + \bar{S}g_\theta\big(\boldsymbol{h}[k]\big), \\ \boldsymbol{h}[0] &= 0, \quad \boldsymbol{h}[i+1] = r_\theta\Big(\begin{bmatrix}\boldsymbol{y}[k-\ell+i] & \boldsymbol{u}[k-\ell+i]\end{bmatrix}, \boldsymbol{h}[i]\Big), \\ \boldsymbol{y}[k+1] &= g_\theta\big(\boldsymbol{h}[\ell], \boldsymbol{u}[k+1]\big), \end{cases} \tag{4-24}$$

with g_θ being an MLP and r_θ is a shorthand notation corresponding to the classical update equations of GRUs (Cho et al., 2014). For simplicity, and as usually done, gates have been omitted from the notation. The encoder and decoder functions are modeled with MLPs. Hyper-parameters and training details[1] are available in appendix A.2.

4.4 Experimental results

We illustrate and evaluate the proposed nonlinear dynamical model identification approach on the estimation and prediction of the state of systems with unknown dynamics. To demonstrate the practical feasibility of our model, we propose to study its behavior in three different

[1] Code and datasets: https://github.com/SteevenJanny/CanonicalStateSpace

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

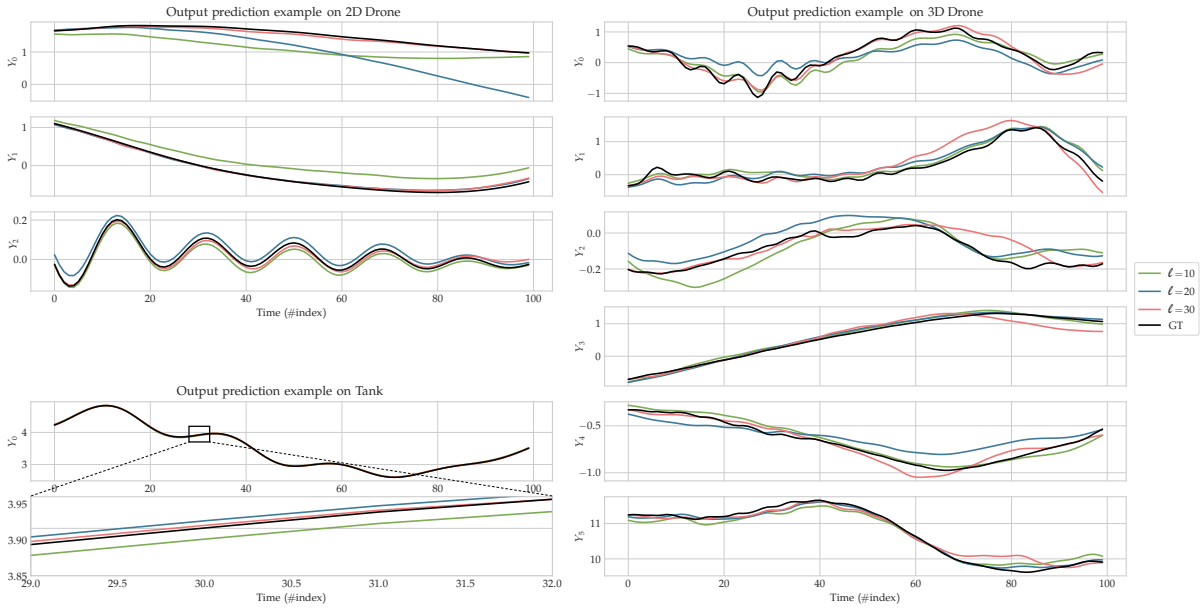


Figure 4.2: **Qualitative results** – Visual example of the output prediction produced by H_θ (*Ours (MLP)*) for different values of ℓ on the three datasets.

scenarios. First, we demonstrate the capabilities of our regressor function H_θ for output prediction on simulated systems. We also study the influence of key parameters, namely the length ℓ of the time window and the impact of the state reduction.

4.4.1 *Dynamical systems and benchmarks*

We use two simulated and one real system to validate our contributions.

- **Tank** – we test the proposed method on the cascade tank system introduced in Schoukens and Noël (2017). This system relates the water level in two connected tanks without consideration of overflow. It has the form off system (4-1) with f and h instantiated as follows

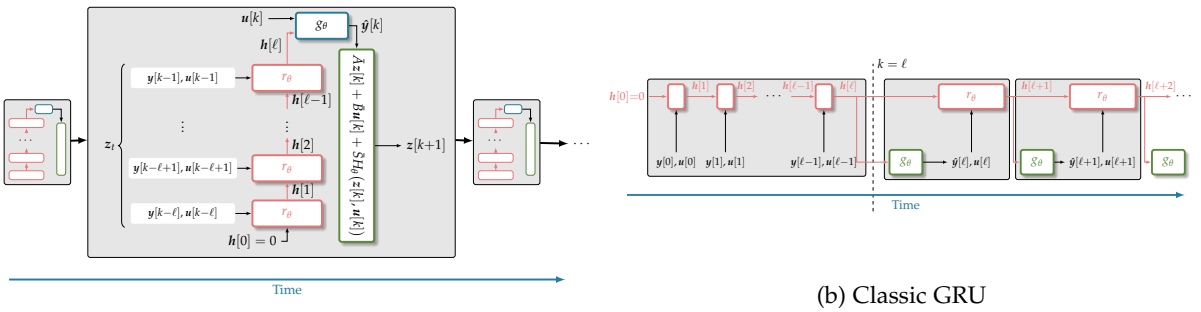
$$\begin{cases} s_1[k+1] &= s_1[k] - k_1\sqrt{s_1[k]} + k_2 u[k], \\ s_2[k+1] &= s_2[k] + k_3\sqrt{s_1[k]} - k_4\sqrt{s_2[k]} \\ y[k] &= s_2[k], \end{cases} \tag{4-25}$$

  with $\boldsymbol{s}[k] = \begin{bmatrix} s_1[k] & s_2[k] \end{bmatrix}^\top \in \mathbb{R}^2$ being the state and k_i, $i = 1, \ldots, 4$ known parameters.

- **2D Drone** – we introduce a model of a 2-dimensional drone, i.e. an UAV moving in a 2D plane. The drone is equipped with two propellers and its dynamic is modeled by:

$$\begin{cases} \ddot{p}_x &= -\frac{k_T}{m}(\Omega_1^2 + \Omega_2^2)\sin(\theta) - \frac{\gamma}{m}(\Omega_1 + \Omega_2)\dot{p}_x, \\ \ddot{p}_z &= \frac{k_T}{m}(\Omega_1^2 + \Omega_2^2)\cos(\theta) - \frac{\gamma}{m}(\Omega_1 + \Omega_2)\dot{p}_z - g, \\ \ddot{\theta} &= \frac{k_T L}{J}(\Omega_2^2 - \Omega_1^2), \\ y &= \begin{bmatrix} p_x & p_z & \theta \end{bmatrix}^\top, \end{cases} \tag{4-26}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) Ours (GRU)

(b) Classic GRU

Figure 4.3: **Comparison with the GRU baseline** – Our model differs from the classic GRU baseline in the definition of the state: (a) our approach defines the state of the system as $z[k]$. The gated recurrent unit is reinitialized at $\boldsymbol{h}[0] = 0$ at each timestep and is only used to aggregate measurements from $z[k]$. (b) Conversely, the GRU baseline uses directly the hidden vector $\boldsymbol{h}[k]$ as the state representation, which is initialized only once at zero, and updated throughout the simulation. For $k > \ell$, the baseline switches to auto-regressive forecasting.

where (p_x, p_z) is the position, θ the angle, k_T the thrust constant, Ω_i the rotationnal speed of the ith propeller, L the length of the UAV, m its mass, J its inertia and γ its friction coefficient. The main interest of such a system is its naturally unstable dynamics, which complicates the identification process. The system has been discretized.

- **3D Drone** – we also evaluate on recordings of the Blackbird UAV flight dataset (Antonini et al., 2018), which consists of 10 hours of aggressive quadrotor flights, measured with an accurate motion capture device. We processed the raw data gathered from the onboard inertial measurement unit (IMU) and propeller rotation speeds as observation and command signals. The regressor is trained to simulate the IMU measurements, i.e. acceleration and angular speed of the drone expressed in the local frame.

Noise has been added to the observations for the two simulated systems, Tanks and 2D Drone. More details about the dataset generation are provided in the appendix A.3.

4.4.2 *Baseline methods*

To experimentally compare our model to competing approaches from the literature, we introduce a neural baseline in the form of GRU (Cho et al., 2014), the state-of-the-art variant of recurrent neural networks. This is a pure data-driven technique from the machine learning field, where the learned state representation directly corresponds to the hidden state vector of the GRU. For a fair comparison, we limit the size of the hidden vector to fit the corresponding size of $z[k]$. We refer to this model as *Classic GRU*. Its update equations are given by

$$\text{Classic GRU:} \quad \begin{cases} \boldsymbol{h}[k+1] &= r_\theta\Big(\begin{bmatrix}\boldsymbol{y}[k] & \boldsymbol{u}[k]\end{bmatrix}, \boldsymbol{h}[k]\Big), \quad \boldsymbol{h}[0] = 0, \\ \boldsymbol{y}[k] &= g_\theta\big(\boldsymbol{h}[k]\big), \end{cases} \tag{4-27}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

ℓ	Tank ($\times 10^{-4}$)				2D Drone ($\times 10^{-2}$)				3D Drone ($\times 10^{-2}$)			
	Class. GRU†	Masti et al.	**Ours (GRU)**	**Ours (MLP)**	Class. GRU†	Masti et al.	**Ours (GRU)**	**Ours (MLP)**	Class. GRU†	Masti et al.	**Ours (GRU)**	**Ours (MLP)**
5	163	1030	138	**7.14**	62.8	60.5	106	**31.4**	24.8	14.6	**6.44**	15.2
10	41.7	1070	5.60	**0.930**	82.7	58.6	68.9	**9.95**	23.7	14.5	**5.32**	14.2
15	4.57	957	3.06	**0.960**	61.9	58.2	35.2	**7.52**	23.4	13.0	**5.07**	13.6
20	4.04	914	1.07	**0.761**	78.4	55.3	19.3	**8.06**	22.7	12.6	**4.68**	13.5
25	0.600	915	**0.481**	0.606	80.3	53.6	23.0	**5.17**	21.3	12.1	**4.83**	13.6
30	1.73	917	**0.193**	0.448	104	51.3	25.0	**3.13**	19.2	12.6	**4.61**	13.1

† The size of the hidden state of each GRU model is adapted to the window size s.t. fits the size of the equivalent regressor model.

Table 4.1: **Quantitative results** – we report MSE error over 100-step rollouts by the learned regression model and compare with baselines, for different window sizes ℓ. Our model consistently outperforms all baselines.

where $r_\theta(\cdot)$ is a gated recurrent unit and $g_\theta(\cdot)$ is a MLP. The baseline is evaluated in a setting that is comparable to the proposed model. In particular, the model has access to the same window of input/output pairs $\left[\boldsymbol{y}[k+i], \boldsymbol{u}[k+i]\right]_{i=1..\ell}$ during the initial burn-in phase. However, these values do not explicitly make up the state, as in our model. This data-driven baseline is sufficiently general to be able to learn the same state representation in theory, but there is no guarantee that training will lead to this solution. The difference between *Ours (GRU)* and this baseline is pictured in figure 4.3

We also experiment with a *latent dynamics* model, in particular, the one introduced in Masti and Bemporad (2018). This model has been evaluated on the same tank system, yet, with a different data collection technique. Train and test trajectory in the Tank dataset as proposed in Masti and Bemporad (2018) are generated from PRBS-like signals, which is a classical approach for system identification. Our version of the Tank dataset is much more challenging: observations are collected from closed-loop simulations with targets generated procedurally and PID control. In our dataset, we took care to explore a wide range of possible states with sparse measurements in the train set to prevent over-fitting on a specific command design.

4.4.3 *Output prediction and parameter analysis*

Output forecasting – the identified dynamical model can be evaluated by performing open-loop forward prediction from initial conditions and the set of inputs applied to the real system. The model then forecasts outputs, which are compared to ground truth measurements. We assessed the first stage of our method using this task, i.e. the resolution of the regression problem. Table 4.1 reports the mean squared error on 100-step roll-out predictions for each baseline and different window sizes $\ell \in \{5, 10, 15, 20, 25, 30\}$. Our method shows excellent prediction error even for low window sizes and consistently outperforms the closest compet-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

ing method from the literature, Masti and Bemporad (2018), by a large margin. We conjecture two key arguments to justify this difference : (1) the structure proposed by Masti and Bemporad (2018) suffers from complex interactions between the auto-encoder and the latent dynamics that penalize learning, and (2) the baseline model over-fitted on the simpler dataset used in the original paper.

Machine learning baseline – is competitive with our contribution. However, its structure forces to observe only one couple $(\boldsymbol{y}[k], \boldsymbol{u}[k])$ at a time. Relevant information needs to be stored in its memory, the hidden state and this storage process is fully learned by gradient descent, a difficult process. In principle, these models can learn a state representation that is similar or even identical to our designed state map, but there is no guarantee that this representation will emerge. Our state map model can therefore be seen as a form of useful inductive bias for recurrent neural models.

For moderate window sizes, our model benefits from the immediate availability of all the components of $\boldsymbol{z}[k]$ in its state. For very large window sizes or complex dynamical systems (such as 3D Drone), the GRU extension (*Ours (GRU)*) outperforms the MLP regressor. In this situation, the extended GRU takes advantage of its incrementally updated memory and manages to manipulate the large dimension of $\boldsymbol{z}[k]$ by processing it piecewise, whereas the MLP must manipulate the entire vector. Figure 4.2 shows samples of the predicted trajectory using the MLP regressor approach for each dataset.

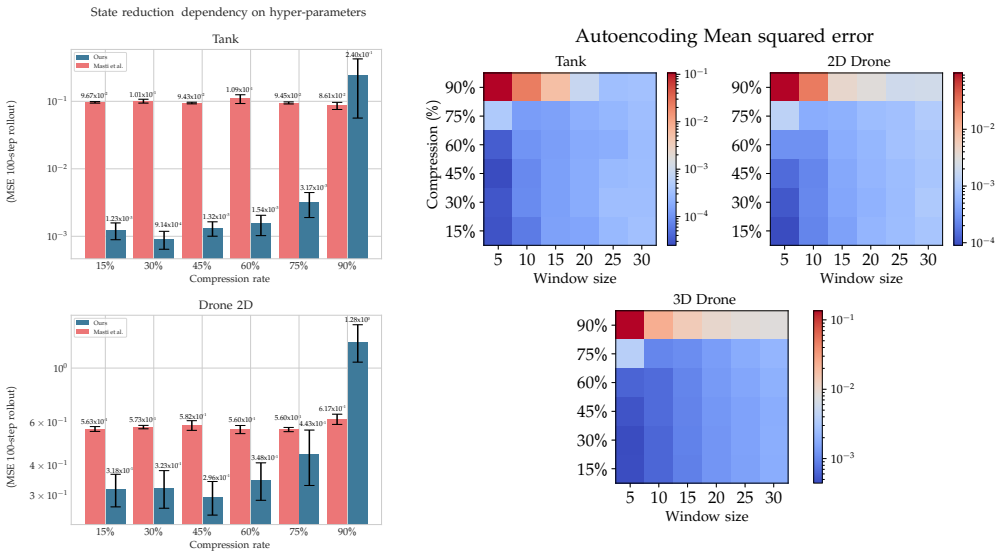
4.4.4 *Model reduction*

The reduction step is performed downstream of the training of the regression model. Nevertheless, the difficulty of the reduction task is directly related to the initial size of the state representation $\boldsymbol{z}[k]$, that is, to the size of the window ℓ. To accurately evaluate the compression capabilities of our approach, we trained several auto-encoders for each value of $\ell \in \{5, 10, 15, 20, 25, 30\}$ corresponding to different rates of compression increasing by steps of 15%.

Figure 4.4b shows the compression capabilities of our encoder-decoder structure for different window sizes ℓ. The results are consistent on the three datasets. The compression rate is more sensitive on small input dimensions, and conversely, a larger dimension can be reduced extensively with negligible loss of accuracy. Indeed, increasing the number of inputs arguably leads to an increase in the redundancies exploitable by the encoder to reduce the dimension of the state space and reconstruct it with limited deviation with respect to the initial vector.

Yet such reduction introduces noise to the state representation that the regressor will have to cope with. We thus evaluate the impact of state-space reduction on the output forecasting capabilities of our model, and summarize the results in figure 4.4a. Our reduction method manages to reduce the dimension of the state in a consistent way up to 60% for the two

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 4.4: (a) We studied the impact of state compression for multiple configurations of final latent state dimension and temporal window and aggregated the results by these two parameters on the synthetic datasets. Specifically, we measure the MSE on prediction for 100 steps in the future (*Ours (MLP)*). (b) Heatmap of MSE for the encoder-decoder structure depending on both the window size (which relates to the initial state dimension dim z) and the compression rate.

datasets in simulation without sensible variation of the prediction error. The error bars reflect the double dependence of our approach both on the performance of the regression model H and on the quality of the encoding-decoding. We compare favorably to the baseline in Masti and Bemporad (2018).

4.5 Conclusion

In this work, we take advantage of the power of high-capacity deep neural networks to design a new methodology for estimating nonlinear dynamical systems from a set of input/output data pairs. We show that the state can be expressed as a state map computed as a function of past inputs and outputs. We learn a mapping from this representation to model outputs from training data using deep networks and show that this approach is competitive.

We tackled the problem of reducing the state space, showing that a state of similar size to the original problem can be obtained through auto-encoding. The methodology has been validated using three numerical examples including a dataset of real-world experiments.

4.6 Post-scriptum: taking a step back

Research has a peculiar dynamics of its own. Every trajectory, regardless of the initial condition, is attracted exponentially fast towards an unstable attractor known as *the submission deadline*. After reaching this point, the dynamics changes and loses its lipschitzness. Trajec-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

tories become chaotic, exhibiting high sensibility to exogenous perturbations (often termed as *reviews*). It is known that there exists a finite time horizon $T_{\text{notification}}$ such that the system becomes quasi-stochastic for all $t > T_{\text{notification}}$, resulting in two separate behaviors, the so-called *acceptance* and *rejected* dynamics. Despite this, the dynamics is globally asymptotically stable, ultimately reaching an equilibrium. In this section, we aim to explore this equilibrium, as well as the insights gathered during this longer convergence horizon. Over time, our own comprehension of our work shifts, benefits from quieter periods and formal or informal discussions. Exciting insights and applications may emerge, motivating future work that, for now, are limited to handwritten notes on a (probably lost) notebook. Of course, we may also discover limitations, or stealth drawbacks that nobody noticed earlier.

Scope of our results – When presenting our work to the community, we encountered two common misconceptions. Firstly, that our model can fully capture the true physical model, with $\bar{\boldsymbol{s}}[k]$ representing the physical state. Achieving this without appealing to additional prior knowledge, such as state supervision or enforcing the exact ODE, is very unlikely. An even more common misinterpretation is that if our model performs well, then unobserved physical quantities must be entangled in $\bar{\boldsymbol{s}}[k]$. This is also untrue. Actually, our method identifies a dynamical model *explaining the observation*, hence modeling solely the *observable part* of the dynamics.

For instance, consider a dynamical model of the form:

$$\dot{\boldsymbol{x}} = f_1(\boldsymbol{x}, \boldsymbol{z}), \quad \dot{\boldsymbol{z}} = f_2(\boldsymbol{z}), \quad \boldsymbol{y} = h(\boldsymbol{z}), \tag{4-28}$$

where the output $\boldsymbol{y}$ can be directly simulated from $\boldsymbol{z}$, making the other part of the state (that is, $\boldsymbol{x}$) unnecessary.

Finally, the learned model is also tied to the control law used for generating the dataset, and might not generalize if the system is controlled with a different algorithm.

About the experiments on the regressor – One of the important aspects of our experiments is that the baselines are constrained to hold the same state size as our method. This creates an asymmetry in the design, particularly visible when comparing *Ours (GRU)* and *Classic GRU*. Let us fix the window length to ℓ so that the state dimension is $L = \dim \boldsymbol{z} = (n_y + n_u) \times \ell$. Naturally, we also constrained the hidden state $\boldsymbol{h}[k]$ in *Classic GRU* (equation (4-27)) to match this dimension. The insight behind this is that we limit the number of information channels to L, i.e. the variables, allowed to flow from one timestep to the next. This addresses a particular case where the state size of the model is expected to be small for some reason. Note that in *Ours (GRU)*, the dimension of the hidden state in the GRU is not limited, since it is initialized to zero at each new iteration.

On one hand, our model cannot freely store information in the state, which is defined canonically as the vector of past measurements. On the other hand, the hidden memory can

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

be larger compared to the baseline (we choose $\dim \boldsymbol{h}[k] = 128$ for our method). This greatly increases the expressivity of our GRU (and also the number of parameters) compared to *Classic GRU*. An unconstrained GRU will likely outperform our model. However, our work shows that limiting the window length in recurrent networks and enforcing physical meaningfulness is beneficial in comparable setups. This is a valuable insight, as one might think about merging both approaches, benefiting from not only the power of GRUs for long-term information, but also from the advantages of $\boldsymbol{z}[k]$, which grounds its behavior in physics.

Determining the state dimension: sufficient vs. necessary – Our model identifies what one might call a *sufficient* window length ℓ. This value can be determined by measuring the error metric for different lengths and keeping the smallest value satisfying some threshold on the accuracy of the regressor. However, not all observations within the resulting vector $\boldsymbol{z}[k]$ might be *necessary*. For instance, several measurements can be redundant or uninformative due to high sampling frequency or noise. As it stands, our method cannot cope with this kind of reduction and is constrained to manipulate the entire window length. Indeed, the compressed state is built using a pure reconstruction task, agnostic of the dynamics. Consequently, we force $\bar{\boldsymbol{s}}[k]$ to contain every information within $\boldsymbol{z}[k]$ including unnecessary observations.

Enhancing the method to truly extract a *necessary* dimension is complex. One promising avenue involves replacing the encoder and decoder with attention modules (Vaswani et al., 2017). The key idea is that attention will *select* certain measurements among $\boldsymbol{z}[k]$ rather than using all of them. This motivates end-to-end training of the dynamics and the auto-encoder, since $\boldsymbol{z}[k]$ could potentially not be reconstructed from $\bar{\boldsymbol{s}}[k]$ anymore. Yet, practical implementation is not straightforward: (1) how can we extract a static state size since the attention weights change dynamically with the inputs (a possible option could be to consider transformers with memory, such as Ryoo et al. (2023)) and (2) how can we handle a varying number of observations when designing the regressors. This is, however, an interesting track to explore.

The "appropriate state size" is a lie – The definition of finite-time observability is binary. Consider a noise-less finite-time observable system over a horizon ℓ^*. If the window length ℓ is smaller than ℓ^*, the system is not observable and the state cannot be retrieved. Conversely, for all $\ell \geqslant \ell^*$, one can theoretically perfectly estimate the true state. We addressed the case of noise in proposition 4.2, but we neglected the effect of learning in the process.

For instance, we observe in our results that the performance of the regressor is improving for $\ell \geqslant \ell^*$. This is reasonable, the model takes advantage of more information. This is somehow related to the evolution of α_ℓ in equation (4-13). We may expect α_ℓ to increase with ℓ, and this evolution is linked to the learning process, hence a direction of future research would be to theoretically study these dependencies with the tool of learning theory (Shalev-Shwartz and Ben-David, 2014). Another interesting direction is to consider the case where $\ell < \ell^*$. In this situation, a trained model might still be able to partially forecast the observations, as the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

observed measurements might still contain regularities that can be exploited. This is more challenging since $\mathcal{O}_\ell$ is not injective anymore.

Finally, defining in practice a minimal window length to guarantee observability is difficult, as the appropriate window length is tailored to the error of a learning model, which, in the general case, will evolve continuously with ℓ. Thus, finding an *appropriate state size* is mainly a matter of user-defined performance requirements.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 5

Deep-KKL: Data-driven Output Prediction for Non-Linear Systems

Work presented at Conference on Decision and Control 2021,
Co-authors: Vincent Andrieu (Université Lyon 1),
Madiha Nadri (Université Lyon 1),
Christian Wolf (NaverLabs Europe)

5.1 Context

5.1.1 *Introduction*

Most of the models introduced so far, including in the related work chapters, capitalize on past measurements to make predictions. For instance, the *latent dynamics* models (introduced in chapter 2) leverage these measurements to estimate a latent state variable. In the previous chapter, measurements are directly integrated into a regressor function for forecasting outputs. We shown that this is connected to the concept of finite-time observability, which pertains to the number of past observations necessary to accurately estimate a state-space representation. A different approach is used in recurrent networks, for instance *Classic GRU* (figure 4.3) in the previous chapter, or PhyDNet discussed p.47. In these methods, the latent state is not estimated in a single shot but rather refined sequentially using the observations within the available time window. These models typically exhibit two modes:

- **The closed-loop mode** starts from a random initial value for the state variable in the latent space. Typically, the latent memory of a recurrent network is initialized at zero. The model is then fed with the observed outputs sequentially and uses them to update the state estimate.
- **The open-loop mode** is used during the prediction step. The model operating in this mode is provided with its own previous predictions as inputs. It becomes autonomous, in the sense that it does not rely on ground truth observations anymore.

However, such a design holds sensible hypotheses. During the closed-loop step, we

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

expect the model to retrieve a good state representation given the measurements. This raises important questions: (1) will the system converge to a good state representation? (2) will it perform reliably for any observed trajectories and more generally (3) what kind of properties guarantee that the closed-loop behaves as expected?

We argue that the answers to these questions lie in the *contraction property*. We intend to delve into how this property can be harnessed within the context of deep learning to ground output prediction into a practical mathematical setup. In this chapter, we develop a framework for designing an output predictor to forecast the observations of an unknown dynamical system. While designed from a control theoretical point of view, it is easily transferable to methods based on deep learning. Moreover, under some assumptions, we show that an upper bound of the prediction error can be computed for predictors complying with our definition.

As a use case of this general approach, we develop an output predictor based on the Kazantzis-Kravaris/Luenberger (KKL) observer (Kazantzis and Kravaris, 1998) for non-linear systems. Building on theoretical work (Andrieu and Praly, 2006; Marconi et al., 2007; Isidori et al., 2010), we develop a data-driven approach to compute a KKL output predictor without any knowledge of the dynamics that generated the observations. Our method mainly relies on deep learning to identify relevant regularities in the training data and extract a predictor from them. We illustrate some of the capabilities of the model across a variety of simulations. We also highlight the limitations, which are due to this constructive solution for KKL. We compare the proposition with two types of deep networks classically used in the field of machine learning for time series forecasting: Recurrent Neural Networks (RNNs) (Rumelhart et al., 1985) and a more modern variant, the Gated Recurrent Unit (GRU) (Cho et al., 2014).

In the same spirit, recent development around the Koopman operator (Lusch et al., 2018; Rowley et al., 2009) proposes to identify a transformation that projects the state of a system into an infinite dimensional latent space, in which the dynamics is fully linear, and then exploits this representation to explain the output. The Koopman operator shares with our work the idea of using deep learning to find a latent representation of a nonlinear system from a set of observed data. Nonetheless, there are a few key differences with our contribution:

- Koopman theory gives an infinite-dimensional transformation into a fully linear system. So any finite-dimensional transformation results as an approximation. By relaxing the constraint on output linearity, KKL guarantees the existence of a finite-dimensional transformation under very weak assumptions.
- The latent space created by the Koopman operator contains information about the full state, while our proposition requires only the *observable* part of the state to be embedded. Thus, our contribution does not require neither a measurement of the complete state nor the observability of the system.
- Koopman requires the mapping from the state to the latent representation and its in-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

verse, whereas KKL only requires the identification of the inverse mapping.

- In contrast to our contribution, methods based on the Koopman operator do not benefit from access to the first steps of the observed trajectory. Their predictions are solely based on the initial state of the system.

5.1.2 *The output prediction problem*

Consider an unknown dynamical system of dimension $n \in \mathbb{N}$ with measured output:

$$\dot{s} = f(s) \ , \ y = h(s) \ , \tag{5-1}$$

with $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ a smooth vector field and $h : \mathbb{R}^n \mapsto \mathbb{R}$ a smooth observation function. For each $s \in \mathbb{R}^n$, we assume that there exists a unique solution to equation (5-1), denoted at time t by $S(s_0, t)$, with s_0 as the initial condition. This solution is defined for all time (i.e. we assume forward and backward completeness). We introduce $\mathcal{Y}$, the set of all possible output functions that can be generated by this dynamical system from the set of initial conditions. Formally,

$$\mathcal{Y} = \Big\{ y : \mathbb{R}^+ \mapsto \mathbb{R}, \exists s_0 \ , \ y(t) = h\big(S(s_0, t)\big) \Big\}. \tag{5-2}$$

The problem we want to solve is:

Given a current time ℓ can we infer the future value of a trajectory y in $\mathcal{Y}$ given that we know $y(t)$, for t in $[0, \ell]$?

Note that we may not solve this problem for all y in $\mathcal{Y}$ but at least for those in a particular subset $\mathcal{Y}$ of $\mathcal{Y}$. We address this problem by first defining a framework encapsulating the observation dynamics into a larger dynamical model, said *generative model* with a *contraction* property. Under some assumptions, we propose an upper bound of the prediction error over time for such a model.

In a second step, we suggest a possible solution via the KKL observer formalism. After proving the existence of a generative model under this particular form, we verify that it also respects the hypothesis required for our upper bound. To demonstrate the feasibility of this solution, and inspired by da Costa-Ramos et al. (2020), we design a learning algorithm to design such KKL model. In our simulations, the KKL-based predictor exhibits remarkable forecasting capabilities, excellent generalization, and good robustness to noise.

5.2 Prediction via embedding into an output-dependent uniform contraction

5.2.1 *Uniform contraction and generating model*

Consider now a dynamical system in the form

$$\dot{z} = G(z, y), \tag{5-3}$$

where z in $\mathbb{R}^m$ and y in $\mathbb{R}$. We denote by $Z(z_0, t, y)$ the solution of equation (5-3) at time t initiated from z_0. This solution depends only on the values of y for t in $[0, \ell]$, i.e. it is causal.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

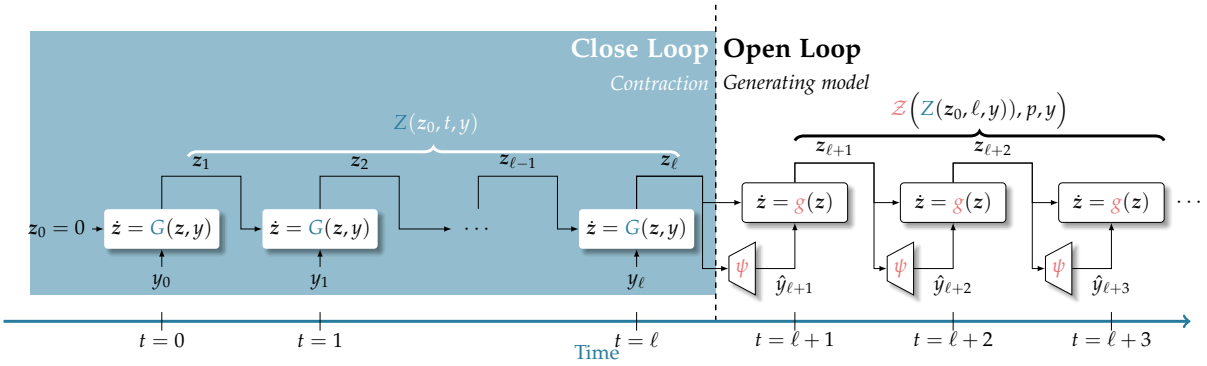


Figure 5.1: **Model overview** – Computation graph for output predictors. The known part of the observation $y(t)$ for $t \leqslant \ell$ is used to make the latent state $Z(z_0,t,y)$ converge to $Z(z_0^y,t,y)$. During the prediction step, we open the loop and let the autonomous system $\dot z = g(z)$ perform the prediction.

Definition 5.1 *(Lohmiller and Slotine, 1998) System (5-3) is said to define a **uniform exponential contraction** if there exist two positive constants k and λ such that for all locally integrable functions $y : \mathbb{R}_+ \mapsto \mathbb{R}$ and all $(z_a, z_b) \in \mathbb{R}^m \times \mathbb{R}^m$ the two solutions $Z(z_a,t,y)$ and $Z(z_b,t,y)$ satisfy:*

$$\big|Z(z_a,t,y) - Z(z_b,t,y)\big| \leqslant ke^{-\lambda t}|z_a - z_b|. \tag{5-4}$$

We are interested in this type of dynamical systems because they *forget* their initial conditions. This will be made precise in Proposition 5.1.

Consider now an autonomous system with measured output

$$\dot z = g(z)\ ,\ y = \psi(z)\ , \tag{5-5}$$

where $g : \mathbb{R}^m \to \mathbb{R}^m$ and $\psi : \mathbb{R}^m \to \mathbb{R}$ and where the solution initiated from z in $\mathbb{R}^m$ and evaluated at time t is denoted by $\mathcal{Z}(z,t)$. Let Y be a subset of $\mathcal{Y}$.

Definition 5.2 *A **generating model** for Y is defined as a couple (g,ψ) such that for all y in Y there exists z_0^y in $\mathbb{R}^m$ such that $y(t) = \psi\big(\mathcal{Z}(z_0^y,t)\big)$.*

For instance, (f,h) is a generating model for the entire set $\mathcal{Y}$. A generating model allows to explain an output y in Y via a dynamical system. If we know the initial condition z_0^y associated with y, future values can be predicted by integrating the generating model from z_0^y.

5.2.2 *Prediction based on contraction and generating model*

We wish to predict the future of any trajectory in $Y \subset \mathcal{Y}$. To this end, the following definition provides two necessary conditions.

Definition 5.3 *An **output predictor** for $Y \subset \mathcal{Y}$ is defined as a couple (G,ψ) such as*

- *$\dot z = G(z,y)$ is a uniform exponential contraction with parameter (k,λ) as in Definition 5.1;*

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

- *the couple* (g, ψ) *with* $g(z) = G(z, \psi(z))$ *is a generating model for* $\mathcal{Y}$.

The behavior of an output predictor is outlined in Figure 5.1. Let ℓ be the number of known timesteps of y and p the number of predicted timesteps. For an output $y \in \mathcal{Y}$, we note z_0^y the exact initial condition such that $\psi(Z(z_0^y, t, y)) = y(t)$ and z_0 the (random) initial condition used in the predictor. The prediction is decomposed into three steps:

1. First, the known part of the observation $y(t), t \in [0, \ell]$ is combined with the contraction property so that $Z(z_0, t, y)$ gets close to $Z(z_0^y, t, y)$. This is the **closed-loop** behavior of the predictor.

2. Then, the autonomous dynamical model $\dot{z} = g(z)$ produces predictions in the latent space $z(t)$, $t \in [\ell, \ell + p]$. We refer to this behavior as **open-loop**, since the real observation y is not used as feedback.

3. Finally, the predicted latent state variables $z(t)$ are inputted to ψ to compute the output $\hat{y}(t) = \psi(z(t))$.

Furthermore, if the dynamics of the latent representation g, and the map ψ are Lipschitz, one can compute an upper bound of the prediction error due to an error on the initial condition z_0.

Proposition 5.1 *Assume there exist* $G : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ *and* $\psi : \mathbb{R}^m \to \mathbb{R}$, *both* C^1, *such that* (G, ψ) *defines an output predictor for* $\mathcal{Y}$ *with:*

$$\left|\frac{\partial g}{\partial z}(z)\right| \leqslant L_1 \,, \quad \left|\frac{\partial \psi}{\partial z}(z)\right| \leqslant L_2 \,, \tag{5-6}$$

with $g(z) = G(z, \psi(z))$, L_1 *and* L_2 *in* $\mathbb{R}^+$, *then for all trajectories* $y \in \mathcal{Y}$, *known in the time interval* $[0, \ell]$, *the prediction* $\hat{y}$ *at the prediction horizon* $p > 0$ *is given as:*

$$\hat{y}(\ell + p) = \psi(\mathcal{Z}(Z(0, \ell, y), p)) \,, \tag{5-7}$$

and satisfies

$$|\hat{y}(\ell + p) - y(\ell + p)| \leqslant k L_2 e^{-\lambda \ell + L_1 p} |z_0^y| \,. \tag{5-8}$$

Proof: See appendix B.1. ■

The prediction mismatch is upper-bounded by a term, which has the following properties:

- As the prediction horizon p increases, the prediction error grows as well. This growth is exponential and depends mainly on the Lipschitz constant of g denoted L_1.
- As ℓ increases, we obtain more information on the output before predicting. For each fixed prediction horizon, the upper bound exponentially goes to zero for increasing p.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

5.3 A possible solution via KKL

5.3.1 *KKL as an output predictor*

In what follows, we derive the KKL observer structure to build an output predictor in the sense of Definition 5.3. For the sake of following mathematical consideration, the state space is reduced to a compact subset $\mathcal{O} \subset \mathbb{R}^n$, and we assume that it is invariant along the dynamics, i.e. for all s_0 in $\mathcal{O}$

$$S(s_0, t) \in \mathcal{O}\ , \forall t\ \in \mathbb{R}. \tag{5-9}$$

We introduce $\mathcal{Y}_\mathcal{O} \subset \mathbb{Y}$, the set of output functions that can be generated by this dynamical system when restricting s_0 to be in $\mathcal{O}$

$$\mathcal{Y}_\mathcal{O} = \left\{ y : \mathbb{R}^+ \mapsto \mathbb{R}, \exists s_0 \in \mathcal{O}\ ,\ y(t) = h\big(S(s_0, t)\big) \right\}. \tag{5-10}$$

Inspired by the KKL observers (Kazantzis and Kravaris, 1998; Andrieu and Praly, 2006), we consider the particular case in which the contracting model given in equation (5-3) is defined on $\mathbb{R}^m$ for some $m \in \mathbb{N}$ and is in the form

$$G(z, y) = Az + by\ , \tag{5-11}$$

with $A \in \mathbb{R}^{m\times m}$ a Hurwitz matrix and $b \in \mathbb{R}^m$ such that (A, b) is a controllable pair. The dynamical model (5-3) with G in equation (5-11) trivially defines a uniform contraction since for all $(z_a, z_b) \in \mathbb{R}^m \times \mathbb{R}^m$ and a given $y \in \mathbb{Y}$

$$|Z(z_a, t, y) - Z(z_b, t, y)| = e^{At}|z_a - z_b|\ . \tag{5-12}$$

Since A is Hurwitz, it yields the existence of k and λ such that equation (5-4) holds. To show that this formalism also defines a generating model, we need to find A, b, and a function ψ such that $\dot{z} = z + b\psi(z)$ generates the output. With the use of Proposition 1, 2, and 3 from Marconi et al. (2007), we have the following statement:

Theorem 5.1 *With $m = 2n + 2$, there exist a Hurwitz matrix A and a vector b with (A, b) controllable and a continuous mapping $\psi : \mathbb{R}^m \mapsto \mathbb{R}$ such that with G defined in equation (5-11), (G, ψ) defines an output predictor for $\mathcal{Y}_\mathcal{O}$ (equation (5-10)).*

Proof: See appendix B.2. ■

Thus, this result confirms that a linear contraction in the form (5-11) may define an output predictor.

Going through the proof, it turns out that almost any complex couple (A, b) of dimension $m' = n + 1$ can be chosen to prove the existence of ψ, as long as A is Hurwitz and (A, b) controllable. One can readily extend the m'-dimensional complex case to our m-dimensional real equation by choosing $m = 2m'$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

5.3.2 *Lipschitz KKL predictor*

The bound on the prediction error obtained in Proposition 5.1 depends on the Lipschitz constants of ψ and g where $g(z) = Az + b\psi(z)$. However, the mapping ψ obtained from Theorem 5.1 may not be globally Lipschitz. In Isidori et al. (2010) sufficient conditions have been obtained to construct a global Lipschitz mapping ψ based on geometric observability assumptions. Inspired by the result obtained in Andrieu (2014) it can be shown that when the dynamical system to predict is observable, a global Lipschitz mapping ψ may be obtained. Consequently, Proposition 5.1 may be employed.

Proposition 5.2 *Assume that h is a global Lipschitz mapping. Assume moreover that the following two observability conditions are satisfied.*

- *Backward Distinguishability: for all (s_1, s_2) in $\mathcal{O}^2$ such that $s_1 \neq s_2$, there exists $t \leqslant 0$ such that $h(S(s_1, t)) \neq h(S(s_2, t))$.*
- *Backward Infinitesimal Distinguishability: for all (s, v) in $\mathcal{O} \times \mathbb{R}^n$ such that $v \neq 0$, there exists $t \leqslant 0$ such that*

$$\frac{\partial h(S(s,t))}{\partial s} v \neq 0, \tag{5-13}$$

then there exist a Hurwitz matrix A, a vector b with (A, b) controllable, a mapping $\psi : \mathbb{R}^m \mapsto \mathbb{R}$ and a positive real number L_2 such that

1. *with G defined in equation (5-11) (G, ψ) defines an output predictor for $\mathcal{Y}_\mathcal{O}$;*
2. *the function ψ has bounded derivative, i.e.*

$$\left| \frac{\partial \psi}{\partial z}(z) \right| \leqslant L_2 \ , \ \forall z \in \mathbb{R}^m; \tag{5-14}$$

3. *the conclusion of Proposition 5.1 holds with $L_1 = \|A\| + \|b\| L_2$.*

Proof: See appendix B.3. ■

The assumptions of the former proposition can be weakened by assuming that there exists an (unknown) change of coordinates, such that in such a coordinate system (5-1) takes a the triangular form

$$\begin{cases} \dot{s}_1 = f_1(s_1) \\ \dot{s}_2 = f_2(s_1, s_2) \end{cases} , \quad y = h(s_1) \ , \tag{5-15}$$

for which the couple (f_1, h) satisfies the observability assumptions of Proposition 5.2. In that case, the former proposition may be applied. Assuming the existence of this change of coordinates is very similar to the assumptions made in Isidori et al. (2010) to obtain that this mapping is globally Lipschitz.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

5.4 Learning KKL with deep networks

In what follows, we propose a constructive method to find ψ based on deep learning. We suppose to have access to two different types of data: (i) during a training phase, we have access to a representative training set of sample trajectories $\mathcal{Y}_D \subset \mathbb{Y}$ to learn $\psi_\theta(z)$, where we now have made explicit in the notation the dependency of ψ on learned parameters θ; (ii) for each trajectory, as described in the previous sections, we have access to the initial output trajectory $y(t) \in \mathcal{Y}$ for $t < \ell$, and are required to forecast the future output up to time $\ell + p$ (where p is the prediction horizon).

5.4.1 *Architecture and training*

We model function ψ_θ as an MLP where θ in $\Theta \subset \mathbb{R}^q$ is the set of parameters to be learned. This class of models is known to have universal approximation power under mild conditions either for infinitely wide (Hornik, 1991) or infinitely deep (i.e. layered) (Lu et al., 2017) architectures, and they also admit Lipschitz constants (Bartlett et al., 2017; Scaman and Virmaux, 2018).

Since the previous section proves the existence of ψ_θ regardless of the choice of $(A, \boldsymbol{b})$ (as long as A is Hurwitz), we decided to learn A freely and fix $\boldsymbol{b} = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^\top$, which reduces the number of degrees of freedom of the model. All parameters are trained by gradient descent to minimize

$$(\theta, A) = \arg\min_{\theta, A} \sum_{y \in \mathcal{Y}_D} \sum_{t=0}^{\ell+p} \left| y(t) - \psi_\theta\big(z(t)\big) \right|^2 \text{ s.t. } \dot{z}(t) = \begin{cases} Az(t) + \boldsymbol{b}y(t) \text{ if } t < \ell, \\ Az(t) + \boldsymbol{b}\psi_\theta(z(t)) \text{ else.} \end{cases} \tag{5-16}$$

For the sake of implementation simplicity, we used a discrete formulation of the dynamics for our simulations using an Euler scheme.

5.4.2 *Data-sets and baselines*

We compare our proposition to two classical types of deep networks designed for time series, namely Recurrent Neural Networks (Rumelhart et al., 1985) with matrix parameters W_1, W_2 and vector $\boldsymbol{b}$

$$z[k+1] = \tanh\left(W_1 z[k] + W_2 y[k] + \boldsymbol{b}\right) \tag{5-17}$$

where $k \in \mathbb{N}$ is a time index such as $y[k] = y(k\delta t)$ for a sampling timestep δt. We also compare with Gated Recurrent Units (Cho et al., 2014)

$$\begin{cases} \boldsymbol{r}[k+1] &= \sigma\big(W_{r1} y[k] + W_{r2} z[k] + \boldsymbol{b}_r\big), \\ \boldsymbol{x}[k+1] &= \sigma\big(W_{x1} y[k] + W_{x2} z[k] + \boldsymbol{b}_x\big), \\ \boldsymbol{n}[k+1] &= \tanh\Big(W_{n1} y[k] + \boldsymbol{r}[k+1] \odot \big(W_{n2} z[k] + \boldsymbol{b}_{n2}\big) + \boldsymbol{b}_{n1}\Big), \\ z[k+1] &= \big(1 - \boldsymbol{x}[k+1]\big) * \boldsymbol{n}[k+1] + \boldsymbol{x}[k+1] * z[k]. \end{cases} \tag{5-18}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Name	Equation	δ_t	$\mathcal{D}$
Van der Pol Oscillator (van der Pol Jun. D.Sc, 1926)	$\begin{cases} \dot{s}_1 &= s_2 \\ \dot{s}_2 &= (1-s_1^2)s_2 - s_1 \end{cases}$	0.25 s	$[-5,5]^2$
Lorenz Attractor (Lorenz, 1963)	$\begin{cases} \dot{s}_1 &= 10(s_2 - s_1) \\ \dot{s}_2 &= 24s_1 - s_2 - s_1 s_3 \\ \dot{s}_3 &= s_1 s_2 - \frac{8}{3}s_3 \end{cases}$	0.02 s	$[-1,1]^3$
Lotka-Volterra Equations (Volterra and Brelot, 1931)	$\begin{cases} \dot{s}_1 &= s_1(\frac{2}{3} - \frac{3}{4}s_2) \\ \dot{s}_2 &= s_2(s_1 - 1) \end{cases}$	0.25 s	$[0,2]^2$
Mean-Field (Noack et al., 2003)	$\begin{cases} \dot{s}_1 &= 0.1s_1 - s_2 - 0.1s_1 s_3 \\ \dot{s}_2 &= s_1 + 0.1s_2 - 0.1s_2 s_3 \\ \dot{s}_3 &= -10(s_3 - s_1^2 - s_2^2) \end{cases}$	0.05 s	$\begin{aligned} s_1 &= r\cos\theta \\ s_2 &= r\sin\theta \\ s_3 &= s_1^2 + s_2^2 \\ \theta &\in [0, 2\pi] \end{aligned}$

Table 5.1: **Description of system used for evaluation** – δt is the final sampling time, and $\mathcal{D}$ the set from where the initial conditions are sampled. For each model, we tried to predict the observation $y = h(\boldsymbol{s}) = s_1$. We used 1000 trajectories for the training sets and 200 for the validation and testing set respectively. These trajectories are generated by solving the differential equation numerically using an RK4 solver with a resolution $10\times$ superior to the final sampling. The observations have been re-scaled so that the training set lies between -1 and 1.

where $\sigma(\boldsymbol{x}) = (1+e^{-\boldsymbol{x}})^{-1}$ is the sigmoid function and $\odot$ is the Hadamard product. These models contain inductive biases in the form of a recurrent memory vector $\boldsymbol{z}[k]$, which allows to propagate the hidden state over time k. In other words, they define latent dynamical systems $\boldsymbol{z}[k+1] = G\big(\boldsymbol{z}[k], y[k]\big)$. The function ψ_θ has the same structure for each of the two variants.

To our knowledge, no proof exists that RNNs and GRUs define proper output predictors in the sense of Definition 5.3. Depending on the learned matrix W_1, the function learned by the RNN may define a contraction (since tanh is monotonic), but there is no rigorous proof that ψ exists for this formalism.

We evaluate our proposition on four different problems that exhibit chaotic behavior: *Van Der Pol* oscillator (van der Pol Jun. D.Sc, 1926), *Lorenz* attractor (Lorenz, 1963), *Lotka-Volterra* predation equations (Volterra and Brelot, 1931) and a *Mean Field* model (Noack et al., 2003) for a fluid flow past a cylinder. Details on these systems are summarized in table 5.1. The interest we have in chaotic systems is that they are highly sensible to error on initial conditions, making them difficult to forecast.

Practically, ψ_θ is an MLP with 3 hidden layers of 128 neurons each. We used ReLU activation functions. Canonically, the dimension of the latent space is equal to $2n+2$ where n is the dimension of the system. Each model is trained with Adam optimizer for 800 epochs, with 64 trajectories per batch. The learning rate is set to 10^{-4}. During training, the model takes as input the ℓ=25 first time steps of the output and outputs the p=25 following time step. Hyper-parameters were optimized over the validation set. For testing, we reduced ℓ to

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

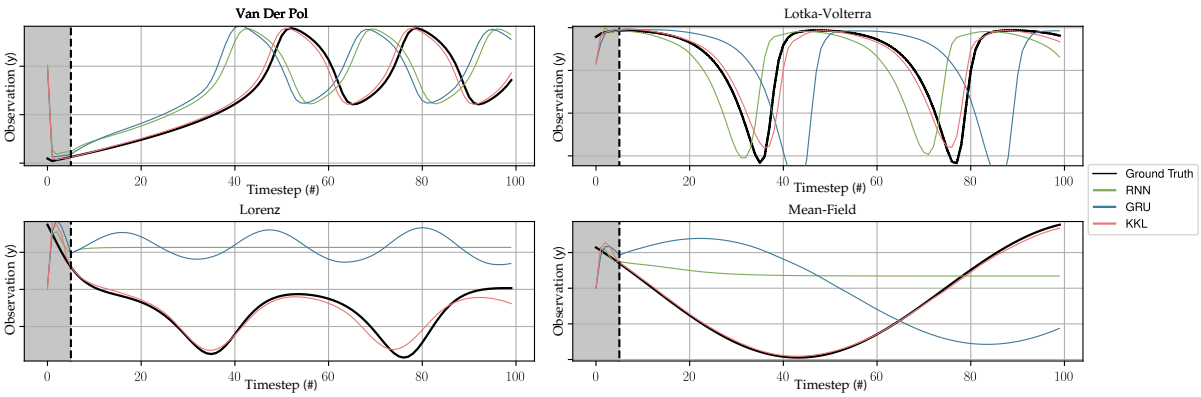


Figure 5.2: **Qualitative results** – Demonstration of output prediction on four nonlinear systems. The $\ell = 5$ first time steps (before the vertical black line) were used during the closed loop step of each model, then the open-loop predicts the $p = 95$ next measurements

	RNN	GRU	Deep-KKL
Van Der Pol	0.0057	0.0343	**0.0013**
Lotka-Volterra	**0.0885**	0.1780	0.1064
Lorenz	0.0441	0.0480	**0.0262**
Mean-Field	0.2254	0.2044	**0.0012**

Table 5.2: **Quantitative results** – MSE on testing set with $\ell = 5$ and $p = 95$. The accuracy of Deep-KKL is at least equal to those of the classic GRU and RNN.

5 time steps and increased p to 95.

5.5 Numerical simulations

5.5.1 *Global performances*

Table 5.2 reports the Mean Squared Error (MSE) on prediction for each model on all four datasets, namely:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{Np} \sum_{y \in \mathcal{Y}_T} \sum_{k=\ell}^{\ell+p} \big(y[k] - \hat{y}[k]\big)^2, \tag{5-19}$$

where $\mathcal{Y}_T$ is the test set of trajectories, of cardinality N. To evaluate the temporal generalization capacities of all models, they were evaluated on a more difficult task than the one they were trained on. They were trained on predicting p=25 future measurements by exploiting ℓ=25 previous measurements. However, during testing, the MSE of Table 5.2 was calculated over p=95 predictions after having seen only ℓ=5 initial time steps. The results show that Deep-KKL generalizes efficiently over this broader horizon, despite the drastic decrease in the amount of data supplied as input (see Figure 5.2).

On our test systems, the accuracy of Deep-KKL is at least equal to those of the classic GRU and RNN, despite its inherent simplicity. By our simulations, we show that Deep-KKL is efficient for output prediction on systems of small dimension while offering a structure more

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

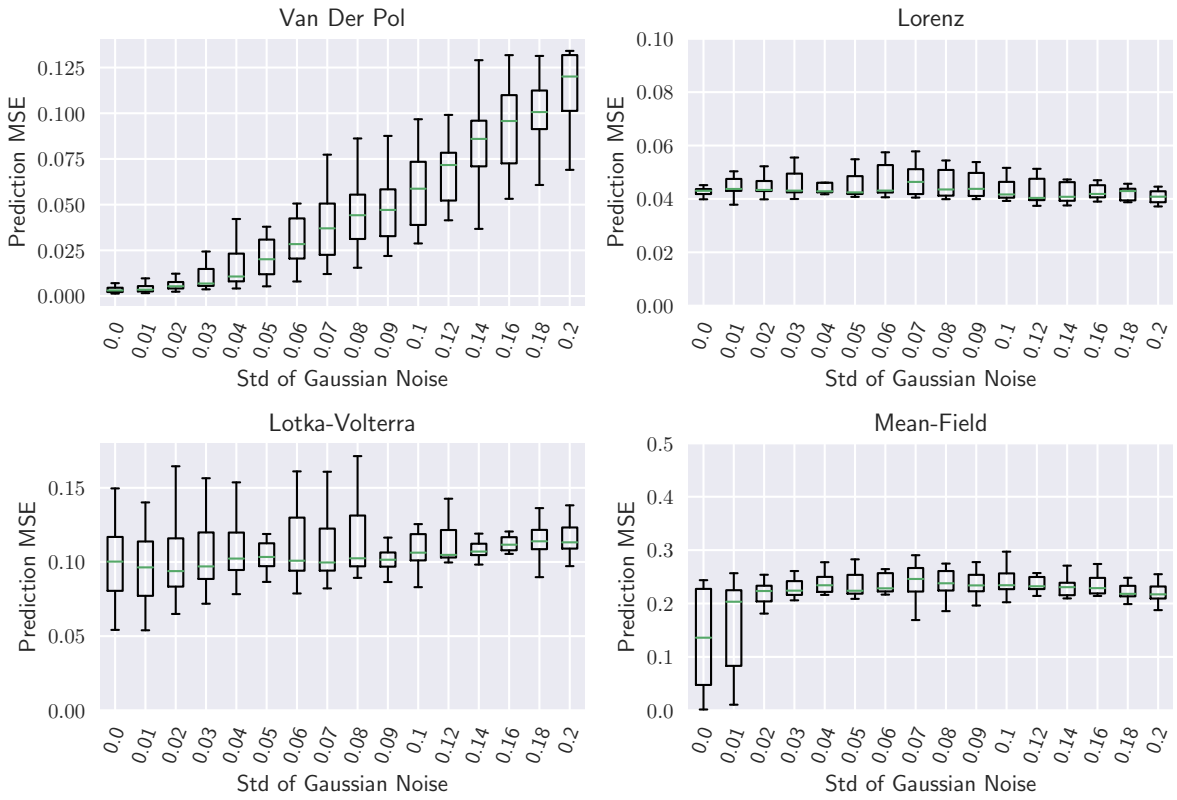


Figure 5.3: **Robustness to noise** – Boxplot of MSE on the test set Y_T according to the amount of noise added during training. Observation measurements lie in $[-1, 1]$. Deep-KKL is capable of handling a reasonable amount of noise in the training data.

suitable for the elaboration of guarantees. Nevertheless, in practice, the recurrent baselines are rarely used in their simple form and are generally stacked, i.e. multi-layered, where one layer takes as input the state of the previous layer. We do not claim, that on systems with very complex dynamics (stochasticity/uncertainty, large dimensions, strong nonlinearity, etc.) Deep-KKL will be competitive with more complex and expressive models (Bézenac et al., 2019; Baradel et al., 2020). However, in our examples, Deep-KKL takes advantage of its simpler structure and manages to perform better. This seems to indicate that for systems of moderate complexity, the use of high-capacity deep models does not seem to be a guarantee of better results.

5.5.2 *Noise Robustness*

In an experimental setup, measurements are inevitably disturbed by noise and errors, either due to mechanical disturbances on the systems or electronic noise associated with the measurements, etc. We decided to evaluate these settings by training our model on noisy observations. In practice, we altered the measured output $y \in Y_D$ with Gaussian noise of zero mean and varying standard deviation.

Figure 5.3 shows the evolution of prediction error made by Deep-KKL as a function of the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

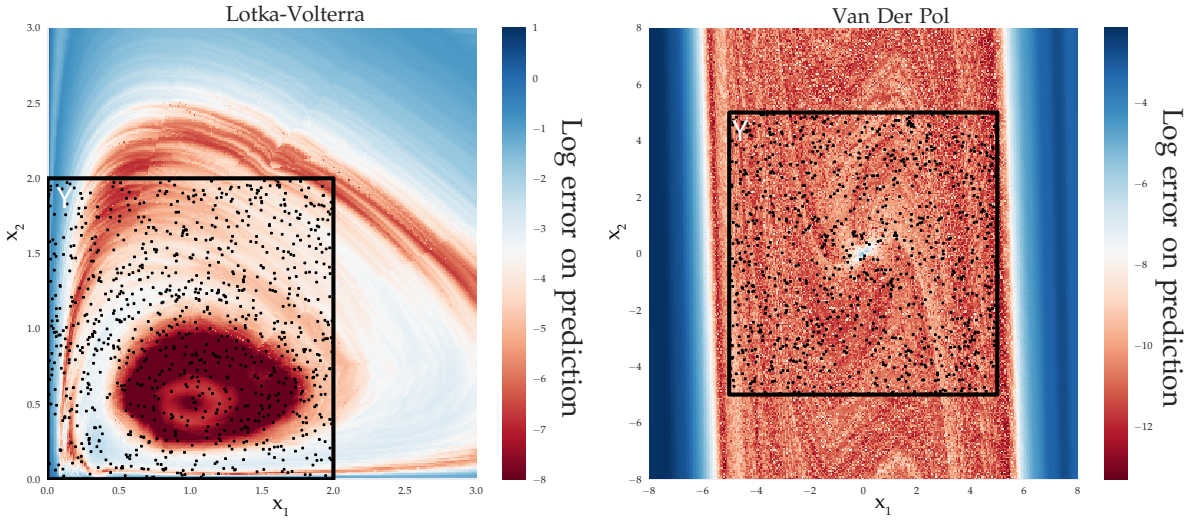


Figure 5.4: **Generalization on unseen domain** – of Deep-KKL for *Van Der Pol* and *Lotka-Volterra* equations. Each dot represents log-MSE on a trajectory starting from the corresponding initial condition $(s_1\ s_2)$. The black square represents the domain of the training set, training trajectories are black dots.

amount of noise added to the training set. Our proposed method is still able to learn with a reasonable amount of noise on the training data.

5.5.3 *Limitations due to Learning*

On top of the initialization error detailed in Proposition 5.1, using deep learning implies another source of error due to the fact that, for a given θ in Θ, the estimation ψ_θ is merely an approximation of the true ψ on $\mathcal{Y}$, which leads to errors in the open-loop phase of the prediction process. The universal approximation theorem of neural networks (Csáji et al., 2001) guarantees that if we allow the set of necessary parameters to be arbitrarily large, then for an arbitrary choice of a constant $\delta > 0$, there exists a set of parameters θ in Θ such that

$$|\psi(z) - \psi_\theta(z)| \leqslant \delta\ , \quad \forall z \in \mathbb{R}^m. \tag{5-20}$$

The evaluation of the constant bound $\delta > 0$ is difficult since we do not have access to the ground truth ψ. The errors $|\psi(z) - \psi_\theta(z)|$ can have multiple reasons, and we will here ignore aspects of learnability (Valiant, 1984), and concentrate on how a given error obtained by ψ_θ impacts the prediction error over time. We formalize this as the following proposition.

Proposition 5.3 *Consider $Y \subset \mathcal{Y}$. Assume that (A, b, ψ) exists such that (G, ψ) with G defines in equation (5-11) is a KKL output predictor for Y. Assume moreover that*

$$\left|\frac{\partial \psi}{\partial z}(z)\right| \leqslant L_2\ . \tag{5-21}$$

and that θ in Θ and $\delta > 0$ satisfy equation (5-20). Then for all trajectories $y \in Y$, known in the time interval $[0, \ell]$, a prediction $\hat{y}_\theta$ at the prediction horizon $p > 0$ given as

$$\hat{y}_\theta(\ell + p) = \psi_\theta\big(\mathcal{Z}_\theta(Z(0, \ell, y), p)\big)\ , \tag{5-22}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

where $\mathcal{Z}_\theta(z_0, p)$ *is the solution initiated from* z_0 *at time* p *of*

$$\dot{z}_\theta = Az_\theta + \boldsymbol{b}\psi_\theta(z_\theta)\,, \tag{5-23}$$

satisfies

$$|y(\ell+p) - \hat{y}_\theta(\ell+p)| \leqslant kL_2 e^{-\lambda\ell + L_1 p}|z_0^y| + \delta\left(L_4 e^{\frac{L_2\|b\|_P}{L_4}p} + 1\right), \tag{5-24}$$

for some positive numbers k, λ, L_1, L_4 *depending on* L_2, A *and* b.

Proof: See appendix B.4. ■

We complete this theoretical analysis by an experimental evaluation, in particular visualization of the generalization capabilities of our model. A central question in machine learning is how a model can generalize from the data it has seen during training, and thus how it performs on unseen data. The distinction between ID (in-distribution) and OOD (out-of-distribution) cases is of particular interest, the latter describing the performance of the model on samples taken from spaces unseen during training. We explore this question and visualize the behavior of Deep-KKL on a larger domain than the set from which the training trajectories have been sampled.

In Figure 5.4, we compute the Log-MSE of Deep-KKL on a grid of trajectories from the *Van der Pol* oscillator and *Lotka-Volterra* equations. For each point on the heat map, we generate the true trajectories from the corresponding initial condition $s(t{=}0){=}(s_1 \quad s_2)^T$ by integrating the corresponding ODE. Then, we use Deep-KKL to predict the output of this system and compare the trajectories. The black square represents the set from which the trajectories in $\mathcal{Y}_D$ were sampled.

There is evidence for excellent in-distribution generalization, as Deep-KKL generalizes well inside the set parameter space covered by Y_D, of course beyond the samples of Y_D themselves. However, we observe limited, but not full OOD generalization, with failure cases when certain parameters are extended beyond the range seen during training.

5.6 Conclusion

We have proposed a theoretical framework for predicting the output of dynamical systems, making it possible to easily define a model capable of representing the dynamics of the observations, and resting solely on two properties. Our proposal is illustrated in a KKL observer combined with learning a solution on a subspace of the observation space with neural networks. Our simulations validate our theoretical results, and demonstrate that Deep-KKL is capable of representing the dynamics of chaotic systems of low dimension. However, the use of learning methods inevitably generates a certain error in the estimates of ψ. Therefore, we proposed a quantification of the effect of this error on the predictions over time.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Lat.	RNN				GRU				KKL			
Size	VdP	Lo.	LV	MF	VdP	Lo.	LV	MF	VdP	Lo.	LV	MF
5	0.0064	0.0429	0.1207	0.2542	0.0193	**0.0424**	0.2855	0.2200	**0.0011**	0.0655	**0.0749**	**0.1628**
10	0.0085	0.0403	0.0858	0.1908	0.0052	**0.0367**	0.0935	0.0963	**0.0018**	0.0453	**0.0466**	**0.0430**
20	0.0032	0.0408	0.1049	0.2712	0.0112	**0.0330**	0.1001	0.0794	**0.0020**	0.0442	**0.0396**	**0.0117**
30	0.0041	**0.0401**	0.0997	0.2754	0.0077	0.0458	0.1736	**0.0771**	**0.0016**	0.0596	**0.0794**	0.0924

Table 5.3: **Different state-size** – We compared Deep-KKL with the recurrent baselines with larger latent state size, on all four datasets. While our approach is competitive on small dimensions (close to $2n+2$), the baselines begin to perform better as the size increases.

5.7 Post-scriptum: taking a step back

Deep-KKL was the first released project of this Ph.D. and is arguably the one that benefits from most introspection, feedback from the community, and connected work. In this section, we will discuss recent work that appeared after the publication of ours. We also present insights, and ideas to better highlight the potential of the method.

Follow up and related work – In the KKL framework, nonlinear systems can be analyzed with the tools of linear dynamics. This motivated a lot of attention from the control community. For a theoretical overview of recent results, see Brivadis et al. (2023). In particular, the backward distinguishability guaranteeing the existence of ψ has been relaxed using set-valued KKL observers (Bernard and Maghenem, 2023), and the framework has been extended to time-varying dynamics in Tran and Bernard (2023).

Considering learning and KKL, the technique has been explored for observation purposes rather than output prediction, i.e. to directly estimate $\boldsymbol{s}$ from the observations y. Note that in this case, the KKL remains in closed-loop mode. For instance, Peralez and Nadri (2021) is a concomitant work using an auto-encoder structure to learn the mapping T (equation (B-8) in the appendix) and its inverse in discrete time. The architecture is related to a *latent dynamics*, and the model is trained to minimize a classic reconstruction loss, together with a dynamics loss, which corresponds the the residual of the discretized PDE (B-6) in the appendix. The method has been extended with ensemble learning in Peralez et al. (2022), and to Neural-ODEs in Miao and Gatsis (2023). Finally, Buisson-Fenet et al. (2022) introduces an empirical criterion to calibrate the linear part of the dynamics in the KKL observer.

Deep-KKL vs. recurrent models – Similarly to the previous chapter, we do not claim that Deep-KKL can outperform unconstrained baselines. In this case, the recurrent models are limited to a hidden state size of $2n+2$, to match our condition of existence. However, surprisingly, while Deep-KKL has a simpler structure, it shows competitive results with the recurrent baselines on chaotic datasets, and beyond the training horizon. We interpret this as a call for moderation: on some tasks, one can build powerful deep networks while keeping relatively simple architectures and benefiting from theoretical guarantees. To complete this analysis,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



Figure 5.5: **Evaluation on noisy setup** – (Top line) Robustness against noisy inputs for models trained on perfect data. Deep-KKL shows limited success in handling the perturbation. (Bottom line) The models are trained with noisy measurements (added Gaussian noise with std. corresponding to 5% of the std. measured on the training dataset), which tend to improve robustness, but also degrade the performance.

we measured the performance with different state sizes and reported the results in table 5.3. Deep-KKL is competitive for smaller state sizes, but recurrent baselines outperform our approach as the latent size increases.

About Noise-robustness – The experiment presented in figure 5.3 evaluates the robustness of Deep-KKL against noise in the training set. The metric is computed on raw measurements (without noise) from the test set, while the model has been trained on noisy data. Another interesting experiment consists of looking at how the model behaves when it is trained on perfect data (no noise), but evaluated in a noisy scenario. The results are shown in figure 5.5, and Deep-KKL shows limited success in handling noisy data. A simple trick to improve robustness consists of training the model with a small amount of noise applied to the inputs. This tends to make the predictions more robust to perturbations.

On the stability of recurrent models – Training models over multiple steps of auto-regressive predictions forces to consider the question of the stability of the dynamical model. This issue is rarely addressed in deep learning, and as a matter of fact, appears fairly rarely with very large models. Issues arise when the gradient descent passes through an iteration where the dynamics becomes unstable. In this case, for some inputs, the forecasted trajectory can explode, and the loss with it. This creates spikes in the training loss, that the model may or may not recover from. The problem is usually addressed with symptomatic solutions, such as gradient or learning rate clipping (Ede and Beanland, 2020; Mai and Johansson, 2021; Krueger and Memisevic, 2016).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

In Deep-KKL, this issue can be addressed in a more principled way, since the stability is directly related to the eigenvalues of A in equation (5-11). Empirically, we found that selecting a small enough learning rate is sufficient to safely train the model, but looking into a provably stable training technique for Deep-KKL is an interesting direction. The theoretical derivations are based on the universal approximation theorem and ignore any errors induced by learning from limited amounts of data. Additional work could study these effects using the tools of learning theory (Shalev-Shwartz and Ben-David, 2014).

Finally, Deep-KKL is a simple yet very powerful model and there are arguably a lot of advantages in integrating it into larger architecture. Moreover, the fact that the dynamics is quasi-linear in the latent space allows for interpretation of how the model actually behaves, which may be of interest for obtaining insights on what the model has learned. The output predictor framework is also a promising way of obtaining theoretical guarantees on recurrent models. For instance, proving that GRU and RNN define contractions is a way to justify why such a model can be initialized from a random initial memory vector.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Part III

Differential equations for simulation and control

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

General remarks

In the previous part, we address fundamental properties to design reliable simulators for dynamical systems, that is the finite-time observability and the contraction property. We will now dive into applied methods, building upon these previous results. In particular, we focus on the resolution of physics-related problems taking the form of PDEs. We start by demonstrating how contraction theory (introduced in the previous chapter) can be used for control purposes. We introduce a new approach for output tracking on nonlinear system requiring to solve a challenging PDE which we address with an advanced PINN-like method. In a second time, we progress toward large-scale learning of fluid dynamics, and introduce a new method for simulating physics in a time and space continuous manner, using insights from finite-time observability properties.

Chapter 6: Deep learning-based output tracking via regulation and contraction theory

TLDR; We address output tracking control problems for input-affine nonlinear systems. We design a two-step state-feedback controller including (1) a contraction-based feedback stabilizer and (2) a feedforward action. The method involves the resolution of a challenging PDE which is approached using PINN-like solver. We also provide theoretical guarantees that the controller behaves correctly despite the approximations implied by the use of neural networks.

Chapter 7: Space and time continuous physics simulation from partial observation

TLDR; We focus on computational fluid dynamics and address the shortcomings of a large part of the literature, which are based on fixed support for computations and predictions in the form of regular or irregular grids. We propose a novel setup to perform predictions in a continuous spatial and temporal domain while being trained on sparse observations. We formulate the task as a double observation problem and propose a solution with two interlinked dynamical systems defined on, respectively, the sparse positions and the continuous domain, which allows to forecast and interpolate a solution from the initial condition. Our model not only generalizes to new initial conditions (as standard auto-regressive models do) but also performs evaluation at arbitrary space and time locations.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 6

Deep learning-based output tracking via regulation and contraction theory

Work presented at World Congress of International Federation on Automatic Control 2023,
Co-authors: Samuele Zoboli (Université Lyon 1),
Mattia Giaccagli (Tel Aviv University),

6.1 Context

This chapter focuses on output tracking, which is arguably among the most versatile applications of control theory. The task consists of designing a control action leading the output of a dynamical system to track an arbitrary reference signal. Such a trajectory may be generated from manual design or any external source, depending on the control task. While being fairly simple to address on linear systems (Francis and Wonham, 1976), output tracking remains an open problem for most general nonlinear dynamical systems. In this case, existing approaches either rely on heavy online computation or demand dynamical model knowledge, and canonically address the output tracking problem by exploiting one of the following tools:

1. **Model inversion** looks for an inverse model mapping the current state-target couple to the input transporting the former to the latter. As an example, we point the reader to solutions based on feedback linearization (Isidori, 1995, Chapter 5.2), (Devasia et al., 1996).

2. **Regulation theory** generalizes the linear method. The controller is divided into a dynamical part (*internal model*) and a stabilizer (Isidori and Byrnes, 1990). The internal model includes a generator of the steady-state solution where the tracking error is zero. The stabilizer provides convergence to such a solution (see e.g. Giaccagli et al. (2022a) for constant references). The control law guarantees stability, attractivity, and forward-invariantness of a manifold where tracking is achieved.

3. **Optimization** considers output tracking as an optimization problem, which motivates the use of the corresponding tools, such as the popular Model Predictive Control.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Approaches 1 and 2 are strongly model-dependent, in the sense that small modeling approximation can cause large errors. Related issues can be alleviated with possible countermeasures such as adaptive techniques (Serrani et al., 2001). Unfortunately, these tricks often leverage challenging analytical considerations, as they usually require a well-defined change of coordinates to bring the system into normal form and, in most cases, a minimum-phase assumption. Conversely, (3) is promising in many ways: it can cope with (small) model errors and it requires moderate theoretical analysis to be deployed. However, theoretical guarantees such as stability are challenging to obtain. Hence, existing results are frequently restricted to simpler classes of systems, e.g., linear ones (Chen et al., 2022).

In this work, we develop a solution to the output tracking problem which intertwines techniques from machine learning and control theory. To do so, we propose a neural network-based algorithm whose backbone comes from output-regulation theory. Formally, we propose a two-step controller. First, we estimate the solution of the regulator equations for a given reference signal, resulting in steady-state trajectory $\pi(t)$ and control action $\psi(t)$ minimizing the tracking error. Then, we design a stabilizer making trajectories asymptotically converge to the reference. To this aim, we rely contractive dynamics. Hence, our stabilizer makes the closed-loop a contraction via the results presented in (Giaccagli et al., 2022b) and approximates its analytical solutions with another DNN.

Notations – Throughout this chapter, $|\cdot|$ denotes the Euclidean norm, $\mathcal{B}_\varepsilon \subset \mathbb{R}^n$ denotes the closed ball of radius $\varepsilon > 0$, i.e. $\mathcal{B}_\varepsilon := \{s \in \mathbb{R}^n, |s| \leq \varepsilon\}$. We say that $\zeta : [0, +\infty) \to \mathbb{R}$ is a class-$\mathcal{K}$ function if $\zeta(0) = 0$ and ζ is strictly increasing.

Given a C^1 vector field $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ and a C^1 2-tensor $P : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n\times n}$, the Lie derivative of the 2-tensor P along the vector field f, denoted as $L_f P(s,t)$, is defined as

$$L_f P(s,t) := \mathfrak{d}_f P + P(s,t)\frac{\partial f}{\partial s}(s,t) + \frac{\partial f}{\partial s}^\top(s,t)P(s,t)$$
$$\text{with } \mathfrak{d}_f := \frac{\partial P}{\partial s} f(s,t).$$

We say that a C^1 vector field $g : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ (respectively, a C^1 matrix function $g : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n\times m}$) is a Killing vector field (or that it satisfies the Killing vector property) with respect to P if $L_g P(s,t) = 0$ (respectively, $L_{g_i} P(s,t) = 0$ for all $i = 1, \ldots, m$, with g_i denoting the i-th column of g) $\forall (s,t)$. We say that a function $\omega \in \mathcal{L}_2$ if it is measurable and $\int_0^{+\infty} |w(\tau)|^2 d\tau < +\infty$.

6.2 An introduction to contraction theory

Before diving further into this work, we propose an introduction to the principles of contraction theory. This section does not provide an extensive description of the domain but aims to give the necessary insights to understand the results of this chapter. Readers familiar with contraction theory and control might skip this section.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

We recall that an autonomous dynamical system $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x}, t)$ ($\boldsymbol{x} \in \mathbb{R}^n$) defines a **uniform exponential contraction** (definition 5.1) if there exist two positive constants k and λ such that for all $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ in $\mathbb{R}^n$ where $\boldsymbol{x} \in \mathbb{R}^n$

$$|X(\boldsymbol{x}_a, t) - X(\boldsymbol{x}_b, t)| \leqslant k e^{-\lambda t} |\boldsymbol{x}_a - \boldsymbol{x}_b| \tag{6-1}$$

for $X(\boldsymbol{x}_0, t)$ the solution at time t initiated from an initial condition $\boldsymbol{x}_0$. Contraction can be interpreted geometrically by imagining two trajectories connected by a line. If the system is contractive, we want the line to shrink exponentially with time. We use this insight to derive conditions for (1) verifying contraction on autonomous systems and (2) designing controllers making the closed-loop system contractive. To do so, we will study the rate of change of the length $L(t)$ between these two trajectories, and extract conditions to ensure that

$$\frac{dL}{dt}(t) \leqslant -\lambda L(t). \tag{6-2}$$

6.2.1 *Linear contraction and contractive controller*

Let us begin with a linear system, $\mathfrak{f}(\boldsymbol{x}, t) = A\boldsymbol{x}$ where $A \in \mathbb{R}^{n \times n}$. The distance between two trajectories initiated at $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ is $L(t) = |X(\boldsymbol{x}_a, t) - X(\boldsymbol{x}_b, t)|$. To study the rate of change of this quantity, we equip $\mathbb{R}^n$ with a Euclidean norm defined by the metric matrix $P \in \mathbb{R}^{n \times n}$, i.e. a symmetric positive definite matrix such that $\forall (\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^n$, the scalar product is $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_P = \boldsymbol{u}^\top P \boldsymbol{v}$ and the canonical norm $|\boldsymbol{u}|_P = \sqrt{\boldsymbol{u}^\top P \boldsymbol{u}}$. The rate of change of $L(t)$ is

$$\frac{d}{dt} L(t) = \dot{L}^\top P L + L^\top P \dot{L} \tag{6-3}$$

Since $\dot{L}(t) = |X(\dot{\boldsymbol{x}_a}, t) - X(\dot{\boldsymbol{x}_b}, t)| = AL(t)$, the contraction condition (6-2) is

$$\frac{d}{dt} L(t) = L^\top (A^\top P + PA) L \leqslant -\lambda L^\top P L \tag{6-4}$$

$$\Leftrightarrow A^\top P + PA \preccurlyeq -\lambda P \tag{6-5}$$

This condition is known as the *algebraic Riccati inequality* and frequently appears in control. Note that it can also be written as $L_{\mathfrak{f}} P \preccurlyeq -\lambda P$ since P is a constant matrix. Moreover, with $P = I$, the condition can be expressed on $\bar{\lambda}$, the largest eigenvalues of A as $\bar{\lambda} \leqslant -\lambda/2$. In other words, an autonomous linear system defines a contraction if and only if A is Hurwitz, that is, if the system is stable. Contraction and stability are the same concepts for linear systems.

We now consider a non-autonomous linear system $\mathfrak{f}(\boldsymbol{x}, \boldsymbol{u}, t) = A\boldsymbol{x} + B\boldsymbol{u}$ with $\boldsymbol{u} \in \mathbb{R}^m$ and $B \in \mathbb{R}^{n \times m}$. We aim to design a controller such that the closed-loop system becomes contractive. In particular, we look for a controller of the form $\boldsymbol{u} = -\rho B^\top P \boldsymbol{x}$ where $\rho > 0$. The condition (6-2) gives

$$\frac{d}{dt} L(t) = L^\top (A^\top P + PA - 2\rho P B B^\top P) L \leqslant -\lambda L^\top P L \tag{6-6}$$

$$\Leftrightarrow A^\top P + PA - 2\rho P B B^\top P \preccurlyeq -\lambda P \tag{6-7}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

For a formal demonstration of this result, see Liberzon (2011). Finding a suitable controller to make the closed-loop system contractive boils down to finding P, λ, and ρ such that inequality (6-7) is satisfied, and the controller $\boldsymbol{u} = -\rho B^\top P \boldsymbol{x}$ is obtained readily.

6.2.2 *Extension to non-linear system*

We will extend the previous results to non-linear systems. Keep in mind that what follows is not a formal derivation, but rather an introduction to contraction theory. For an in-depth study, see for instance Andrieu et al. (2016); Lohmiller and Slotine (1998); Manchester and Slotine (2017). Directly studying the length $L(t)$ for nonlinear systems is tedious. Instead, we will divide $L(t)$ in small segments so that we can linearize the dynamics. We call $\delta \boldsymbol{x}$ one of these segments, which can be interpreted as the virtual displacement between two neighboring trajectories. The dynamics of $\delta \boldsymbol{x}$ is

$$\dot{\delta \boldsymbol{x}} = \frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}, t)\delta \boldsymbol{x} \tag{6-8}$$

Similarly to the linear case, we can now study how this distance changes. Yet, since the dynamics depends on $\boldsymbol{x}$, a constant metric P is not sufficient. We need to consider a varying metric $P(\boldsymbol{x}, t)$ to account for the variations of $\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}, t)$. In other words, we are shifting from Euclidean metrics to Riemannian ones, allowing for distortion of the state space. The rate of change of the distance $\delta \boldsymbol{x}$ is given by

$$\frac{d}{dt}\delta \boldsymbol{x}^\top P(\boldsymbol{x}, t)\delta \boldsymbol{x} = \delta \boldsymbol{x}^\top \left(\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}^\top P(\boldsymbol{x}, t) + P(\boldsymbol{x}, t)\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}} + \dot{P}(\boldsymbol{x}, t) \right) \delta \boldsymbol{x} \tag{6-9}$$

Since $\dot{P} = \frac{\partial P}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial t} = \mathfrak{d}_\mathfrak{f} P$, the contraction condition is

$$\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}^\top P(\boldsymbol{x}, t) + P(\boldsymbol{x}, t)\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}} + \mathfrak{d}_\mathfrak{f} P(\boldsymbol{x}, t) \preccurlyeq -\lambda P(\boldsymbol{x}, t) \tag{6-10}$$

or simply $L_\mathfrak{f} P \preccurlyeq -\lambda P$. This condition is similar to the one for the linear case, with the exception that it accounts for the dependence of the metric on the state variable.

We now consider an input-affine dynamical system and its extended linearized system

$$\begin{aligned} \dot{\boldsymbol{x}} &= \mathfrak{f}(\boldsymbol{x}, t) + \mathfrak{g}(\boldsymbol{x}, t)\boldsymbol{u} \\ \dot{\delta \boldsymbol{x}} &= \underbrace{\left(\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}, t) + \sum_{i=1}^{m} \frac{\partial \mathfrak{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{x}, t))\boldsymbol{u}_i \right)}_{A(\boldsymbol{x}, \boldsymbol{u}, t)} \delta \boldsymbol{x} + \mathfrak{g}(\boldsymbol{x}, t)\delta \boldsymbol{u} \end{aligned} \tag{6-11}$$

We are looking for a controller $\boldsymbol{u} = \beta(\boldsymbol{x}, t)$ making the closed-loop system contractive, where β is a C^2 function such that $\frac{\partial \beta}{\partial \boldsymbol{x}} = -\rho \mathfrak{g}^\top P(\boldsymbol{x}, t)$. One can verify that we retrieve the state-feedback controller $\beta(\boldsymbol{x}, t) = -\rho B^\top P \boldsymbol{x}$ in the linear case. This design choice already gives a first condition for existence: the function β must be C^2, and $\mathfrak{g}^\top P(\boldsymbol{x}, t)$ must be integrable. We now compute the rate of change

$$\frac{d}{dt}\delta \boldsymbol{x}^\top P(\boldsymbol{x}, t)\delta \boldsymbol{x} = \delta \boldsymbol{x}^T \underbrace{\left(\left(A - \rho \mathfrak{g}\mathfrak{g}^\top P \right)^\top P + P\left(A - \rho \mathfrak{g}\mathfrak{g}^\top P \right) + \dot{P} \right)}_{K(\boldsymbol{x}, t)} \delta \boldsymbol{x} \tag{6-12}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

On the other hand, $\dot{P} = \frac{\partial P}{\partial x}(\mathfrak{f}(x,t) + \mathfrak{g}(x,t)u) = \mathfrak{d}_{\mathfrak{f}}P + \mathfrak{d}_{\mathfrak{g}}Pu$, hence by developing $K(x,t)$

$$K(x,t) = \overbrace{\mathfrak{d}_{\mathfrak{f}}P + \frac{\partial \mathfrak{f}}{\partial x}^{\top}P + P\frac{\partial \mathfrak{f}}{\partial x}}^{L_{\mathfrak{f}}P} - 2\rho P\mathfrak{g}\mathfrak{g}^{\top}P + \sum_{i=1}^{m}\beta_i(x,t)\underbrace{\left(\mathfrak{d}_{\mathfrak{g}_i}P + \frac{\partial \mathfrak{g}_i}{\partial x}^{\top}P + P\frac{\partial \mathfrak{g}_i}{\partial x}\right)}_{L_{\mathfrak{g}}P} \tag{6-13}$$

The closed-loop system is contractive if $K(x,t) \preccurlyeq -\lambda P$. We retrieve the Riccati-like condition $L_{\mathfrak{f}}P \preccurlyeq -\lambda P$ with an additional term $L_{\mathfrak{g}}P$ depending on the command. In the linear case, the action of the command is the same everywhere in the state space, consequently, the distance between two trajectories with the same input and different initial conditions cannot increase because of the input. This is not true in the nonlinear case, since the effect of the command depends on the state. A simple workaround consists in assuming $\mathfrak{g}$ to be a *Killing vector field*[1] for P, i.e. $L_{\mathfrak{g}}P = 0$ (see (Manchester and Slotine, 2017, Section III.A) or (Giaccagli et al., 2022a, Section II.B)). Finally, the existence of the controller is subject to:

1. The existence of a positive definite symmetric matrix $P(x,t)$ and positive constants λ, ρ such that
$$L_{\mathfrak{f}}P - 2\rho P\mathfrak{g}\mathfrak{g}^{\top}P \preccurlyeq -\lambda P \tag{6-14}$$

2. The existence of a C^2 function $\beta(x,t)$ satisfying the integrability condition
$$\frac{\partial \beta}{\partial x} = \mathfrak{g}^{\top}P(x,t) \tag{6-15}$$

3. The vector field $\mathfrak{g}$ satisfies the Killing vector condition
$$L_{\mathfrak{g}}P = 0 \tag{6-16}$$

6.3 Preliminaries

In this chapter, we consider a system of the form

$$\dot{s} = f(s) + g(s)u \tag{6-17a}$$
$$e = h(s) - r(t) \tag{6-17b}$$

where $s \in \mathbb{R}^{n_s}$ is the state, $u \in \mathbb{R}^{n_u}$ is a control action, f, g, h are sufficiently smooth functions and $e \in \mathbb{R}^{n_e}$ is the error between an output $y = h(s)$ and a known smooth time-varying reference $r(t)$ taking values on a compact set $\mathcal{R} \subset \mathbb{R}^{n_e}$. Our goal is to design a feedback control action u such that the error e asymptotically goes to zero.

From a regulation theory viewpoint, output tracking can be achieved if and only if there exist two sufficiently smooth mappings $\pi : \mathbb{R}^{n_e} \to \mathbb{R}^{n_s}$ and $\psi : \mathbb{R}^{n_e} \to \mathbb{R}^{n_u}$ solutions of the

[1] named after Wilhelm Killing

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

so-called *regulator equations* (Isidori and Byrnes, 1990; Byrnes and Isidori, 2003)

$$\begin{aligned} \dot{\pi}(\boldsymbol{r}(t)) &= f\Big(\pi(\boldsymbol{r}(t))\Big) + g\Big(\pi(\boldsymbol{r}(t))\Big)\psi(\boldsymbol{r}(t)) \\ 0 &= h\Big(\pi(\boldsymbol{r}(t))\Big) - \boldsymbol{r}(t)\,. \end{aligned} \tag{6-18}$$

In simple words, the solution (π, ψ) of the regulator equations is an admissible (i.e. realizable) trajectory of the system that produces the desired output reference $r(t)$. The mapping π represents the *steady-state manifold* on which the tracking error is zero, and the mapping ψ is the *steady-state control action* making such a manifold forward invariant along the trajectories of the system.

We look for a controller capable of making the system contractive toward the steady-state manifold trajectory so that any trajectory will converge exponentially to π. The controller has the following form

$$\boldsymbol{u} = \gamma(\boldsymbol{s}, t) = \psi(t) + \alpha(\boldsymbol{s}, \pi(t), t), \tag{6-19}$$

where α is any function that forces the dynamics of $\boldsymbol{s}$ to converge to $\pi(t)$ and is asymptotically vanishing, i.e. $\alpha(\boldsymbol{s}, \boldsymbol{s}, t) = 0$ for all $(\boldsymbol{s}, t) \in \mathbb{R}^{n_s} \times \mathbb{R}$.

For designing the controller, we take inspiration from (Giaccagli et al., 2022b), where the synchronization problem is cast into the contraction framework. The authors provide a constructive design achieving multi-agent synchronization for a network of input-affine time-varying nonlinear systems. The general control structure (6-19) is inspired by the results in (Pavlov et al., 2006, Section 5.4). However, we highlight four main differences in our approach.

1. In (Pavlov et al., 2006, Section 5.4) the authors propose $\alpha = K(\boldsymbol{s} - \pi)$, with K being a constant matrix. In our design, through the notion of "Killing vector field", we provide a more general structure for the controller α.

2. We show that approximate, rather than asymptotic, tracking can be achieved under a non-perfect knowledge of the mappings (π, ψ).

3. We provide a neural networks-based algorithm for the estimation of (π, ψ) (and of the control action α) that generalizes over references.

4. We link the performance of the neural network to the tracking error.

6.4 Main results

We formalize our problem as follows. Let $c \geqslant 0$ and assume to know a smooth function $\gamma : \mathbb{R}^{n_s} \times \mathbb{R} \to \mathbb{R}^{n_u}$ such that the system (6-17) in closed-loop with the feedback control $\boldsymbol{u} = \gamma(\boldsymbol{x}, t)$ has bounded trajectories and such that $\lim_{t\to+\infty} |e(t)| \leqslant c$. Then:

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

- if $c = 0$, we say that the *asymptotic* output tracking control problem is solved;
- if $c > 0$, we say that the *approximate* output tracking control problem is solved.

The goal is to show that (i) under perfect knowledge of the system, of the regulation equations, and of the control structure, the asymptotic output tracking problem is solved; (ii) it is still possible to achieve approximate output tracking using an approached solution. Then, (iii) we show that the tracking error can be linked to the approximation errors of the estimated quantities, and we provide bounds guaranteeing approximate tracking up to arbitrary precision.

6.4.1 *Approximate output tracking: the analytic solution*

We start by assuming the following.

Assumption 6.1 *Consider system (6-17), (6-18) and let $\varphi(s,t) = f(s) + g(s)\psi(t)$. There exist a C^1 matrix function $P : \mathbb{R}^{n_s} \times \mathbb{R} \to \mathbb{R}^{n_s \times n_s}$ taking symmetric positive definite values, a C^2 function $\beta : \mathbb{R}^{n_s} \times \mathbb{R} \to \mathbb{R}^{n_u}$, positive real numbers $\underline{p}, \bar{p}, \varepsilon, \rho > 0$ such that, for all $(s,t) \in \mathbb{R}^{n_s} \times \mathbb{R}$, the following holds:*

1. *The matrix function P satisfies*

$$\begin{aligned} &L_\varphi P(s,t) - \rho P(s,t) g(s) g^\top(s) P(s,t) \preccurlyeq -\varepsilon P(s,t)\,, \\ &\underline{p} I \preccurlyeq P(s,t) \preccurlyeq \bar{p} I\,. \end{aligned} \tag{6-20}$$

2. *The function β satisfies the integrability condition*

$$\frac{\partial \beta^\top}{\partial s}(s,t) = P(s,t) g(s)\,. \tag{6-21}$$

3. *The Killing vector property holds, namely*

$$L_g P(s,t) = 0. \tag{6-22}$$

These assumptions are similar to the one introduced in section 6.2.

Proposition 6.1 *Consider system (6-17), (6-18) and let $\varphi(s,t) = f(s) + g(s)\psi(t)$. Let Assumption 6.1 hold and let $\omega : \mathbb{R} \to \mathbb{R}^{n_u}$ be in $\mathcal{L}_2$. Then, for any $\kappa > \frac{\rho}{2}$, the trajectories of system (6-17) in closed-loop with*

$$u = \psi(t) + \alpha(s, \pi(t), t) + \omega(t) \tag{6-23a}$$

where

$$\alpha(s, \pi(t), t) = -\kappa\big(\beta(s,t) - \beta(\pi,t)\big) \tag{6-23b}$$

satisfy

$$|\mathcal{S}(s_0, t_0, t) - \Pi(\pi_0, t, t_0)| \leqslant k\,|s_0 - \pi_0|\, e^{-\lambda(t-t_0)} + \zeta\big(\,|\omega(t)|\,\big) \tag{6-24}$$

for all $(s_0, \pi_0, t, t_0) \in \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times [t_0, \infty) \times \mathbb{R}$, for some $k, \lambda > 0$ and for some class-$\mathcal{K}$ function ζ, with $\mathcal{S}(\cdot)$ being the trajectory of (6-17) in closed-loop and $\Pi(\cdot)$ the trajectory of (6-18).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Proof: See Appendix C.1. ■

The result of Proposition 6.1 shows that the control law (6-23) guarantees that trajectories of (6-17) in closed-loop remain close to the solution of (6-18). In particular, the error between the two depends on the component $\omega(t)$ in (6-23). Our objective is to approximate the control action with neural networks. Then, in our case, $\omega(t)$ represents an approximation error. Without full knowledge of the model and of $(\pi(t), \psi(t))$ solutions of (6-18), we end up using a control law of the form

$$\boldsymbol{u} = \psi_\theta(t) - \kappa(\beta_\theta(\boldsymbol{s}, t) - \beta_\theta(\pi_\theta, t)), \tag{6-25}$$

where $\psi_\theta, \pi_\theta, \beta_\theta$ represent suitable approximations of the functions ψ, π, β in (6-23) where we already made explicit the dependency on parameters θ. In what follows, we link the error in the control action to the approximation capabilities of our structure. More specifically, we show that if the functions $\psi_\theta, \pi_\theta, \beta_\theta$ are sufficiently close to the true functions, then approximate output tracking is still achieved. This lays strong foundations for the following section, as we exploit neural networks to learn an approximate version of the exact functions, which are not explicitly computable in general. Hence, via Proposition 6.1 and the following result, we highlight the link between the approximation and the tracking error.

Proposition 6.2 *Consider system (6-17) in closed-loop with the control law (6-25). Let (π, ψ) be a solution of (6-18) and let $\varphi(\boldsymbol{s}, t) := f(\boldsymbol{s}) + g(\boldsymbol{s})\psi(t)$. Let (κ, β) be chosen as in Proposition 6.1. Then, for any compact sets $\mathcal{W}_{\tilde{\boldsymbol{s}}} \subset \mathbb{R}^{n_s}$, $\mathcal{R} \subset \mathbb{R}^{n_e}$ such that $r(t) \in \mathcal{R}$ for all $t \geqslant t_0$ and for any $\delta \geq 0$, there exist a compact set $\mathcal{W}_s$ and a scalar $\mu_\delta \geq 0$ such that, if the following holds for all $(\boldsymbol{s}, t) \in \mathcal{W}_s \times [t_0, \infty)$*

$$\begin{aligned} |\beta_\theta(\boldsymbol{s}, t) - \beta(\boldsymbol{s}, t)| &\leqslant \mu_\delta, \\ |\psi_\theta(t) - \psi(t)| &\leqslant \mu_\delta, \\ |\pi_\theta(t) - \pi(t)| &\leqslant \mu_\delta, \end{aligned} \tag{6-26}$$

then

$$\lim_{t \to +\infty} |\mathcal{S}(\boldsymbol{s}_0, t, t_0) - \Pi(\pi_0, t, t_0)| < \delta. \tag{6-27}$$

for any $(\boldsymbol{s}_0, \pi_0)$ satisfying $(\boldsymbol{s}_0 - \pi_0) \in \mathcal{W}_{\tilde{\boldsymbol{s}}}$.

Proof: See Appendix C.2. ■

6.4.2 *DNN-based output tracking controller*

We propose to implement $\psi_\theta, \pi_\theta, \beta_\theta$ using deep learning. Neural networks are typically Lipschitz functions by construction. Hence, if the approximation error over the training dataset is bounded, the error over a compact set including the dataset is also bounded. This allows the application of Proposition 6.2. In what follows, since the time-dependency of φ is due only to the reference $r(t)$ (because of the tracking task), we consider $t_0 = 0$. Thus, we discretize the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

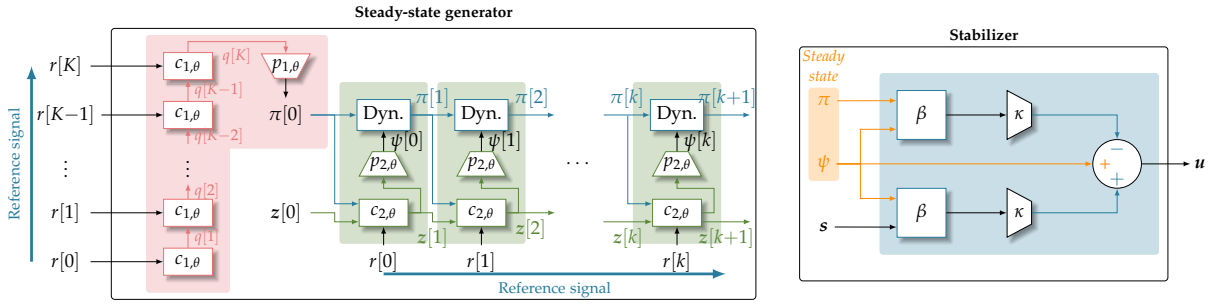


Figure 6.1: **Model overview** – We address the output tracking problem with a twofold approach: the *state reference generator* approximates the solution of the regulator equations and computes states and inputs given an arbitrary reference signal. The *stabilizer* leverages a learned contractive function to force the dynamical system to track the reference.

problem using the Euler scheme with a small timestep τ, yielding $\boldsymbol{s}(k\tau) = \boldsymbol{s}_k$. We organize our algorithm into three steps[2]:

Step 1: Solve the regulator equations

Our goal is to identify a pair of steady state/input (π, ψ) such that the resulting trajectory of the system reproduces the reference signal. In other terms, we are looking for the solution of the regulator equations (6-18). The next steps will focus on building a controller making the closed-loop system contractive with respect to this steady-state, thus achieving output tracking.

We propose a two-step approach: first an initial condition $\pi[0]$ is estimated from the reference signal. Since the plant may not be fully observable from a single point, estimating the initial state can take advantage of longer reference signal $r = \{r[k] \,|\, k \in [0, K]\}$. Then, we simulate the system (6-17) while computing commands $\{\psi[k] \,|\, k \in [0, K]\}$ on the fly using a second neural structure. Formally:

$$\text{Initial state estimation:} \quad \pi[0] = p_{1,\theta}(\boldsymbol{q}[k]) \quad \text{st.} \begin{cases} \boldsymbol{q}[0] & = 0 \\ \boldsymbol{q}[k+1] & = c_{1,\theta}(q[k], r[k]) \end{cases}$$

$$\text{Next control input:} \quad \psi[k] = p_{2,\theta}(\boldsymbol{z}[k]) \quad \text{st.} \begin{cases} \boldsymbol{z}[0] & = 0 \\ \boldsymbol{z}[k+1] & = c_{2,\theta}(\pi[k], r[k+1], \boldsymbol{z}[k]) \end{cases}$$

$$\text{Simulation:} \quad \pi[k+1] = \pi[k] + \tau\Big(f(\pi[k]) + g(\pi[k])\psi[k]\Big)$$

The initial state estimate is obtained by aggregating the reference signal into a unique vector using a GRU $c_{1,\theta}$ followed by an MLP $p_{1,\theta}$. The dynamics is then simulated, using inputs computed at each step using another GRU $c_{2,\theta}$ followed by a MLP $p_{2,\theta}$. Note that the reference signal may change during the interval $[0, K]$. In that case, a new estimate of $\pi[0]$ is obtained

[2]Our code can be found at: https://github.com/SteevenJanny/OutputTracking_contraction.git

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

by running through the state reference generator again. The model is trained to minimize the mean squared error between the discrete reference signal $\{r[k] \mid k \in [0, K]\}$ and the predicted observation $\{h(\pi[k]) \mid k \in [0, K]\}$. Architectures and training details are provided in Appendix C.3. Note that by using the entire reference signal to estimate the initial condition, we make use of the finite-time observability results stated in chapter 4. Indeed, the system might not be observable from a single observation, so the estimation of $\pi[0]$ can benefit from a longer observation window.

Our setup shares similarity with PINNs in the sense that we are seeking for the solution of a known differential equation (the regulator equations). However, in our case, we are solving for two coupled variables π and ψ. Moreover, and conversely to PINNs, the initial condition $\pi[0]$ is unknown and must be estimated beforehand. These particularities force to adapt the architecture accordingly with regards to standard PINN techniques. Importantly, our models can be trained offline and do not need to be re-optimized on the fly. For these reasons, our approach is closer to *learning with a differentiable simulator* rather than a PINN-like setup.

Step 2: Find a suitable metric P

This is arguably the most challenging step of our method. We addressed the problem as a challenging PINN setup, where the solution is a matrix. Our goal is to find a metric $P_\theta \succ 0$ satisfying the following constraints:

- **Symmetric** – this is imposed as a hard constraint by estimating solely the upper triangular part of P_θ,
- Synchronization – is relaxed as a soft constraint and enforced via a loss term,
- Killing vector – is also relaxed with another loss term,
- Positive definiteness – is formulated as a condition on the eigenvalues of P_θ, and enforced via the loss.

The key difficulty of the problem is that the PDE is in fact a matrix inequality. A classic approach could be to discretize the state space and ensure that each constraint is respected at least locally. We propose a different method by formulating each constraint as a negative definiteness condition on four matrices M_i taking the form of

$$\begin{aligned} M_1 &= L_\varphi P_\theta(s,\psi) - \rho P_\theta(s,\psi) g(s) g^\top(s) P_\theta(s,\psi) + \varepsilon I, \\ M_2 &= L_g P_\theta(s,\psi) - \epsilon I, \\ M_3 &= -L_g P_\theta(s,\psi) - \epsilon I, \\ M_4 &= -P_\theta(s,\psi) + \underline{p} I. \end{aligned} \tag{6-28}$$

We supervise negative definiteness on M_i by enforcing the maximum eigenvalue $\lambda(M_i)$ to have a negative real part $\mathcal{R}\{\lambda(M_i)\}$. Since our constraints are formulated as inequalities, the problem is solved when every eigenvalue has a negative real part, which motivates the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

following loss

$$J_{P,1}(\boldsymbol{s},\psi,\boldsymbol{p}) = \sum_{i=1}^{4} w_i J_i(\boldsymbol{s},\psi,\boldsymbol{p}), \text{ with } J_i(\boldsymbol{s},\psi,\boldsymbol{p}) = \ln\left(\max\left(\mathcal{R}\{\lambda(M_i)\},0\right)+1\right), \tag{6-29}$$

where $\boldsymbol{p} = (\rho,\varepsilon,\epsilon,\underline{p})$ is a set of learned parameters and $w_{P,1} := (w_1,\ldots,w_4)$ is a vector of (positive) scalar weights. The interest we have in this loss is that it reaches zero when all conditions are satisfied. The parameter vector $\boldsymbol{p}$ controls the margins on the matrix inequalities. We propose a modified objective for training the metric function P_θ and estimating $\boldsymbol{p}$ using a switching loss function, composed of two interacting elements

$$J_P(\boldsymbol{s},\psi,\boldsymbol{p}) \quad = J_{P,1}(\boldsymbol{s},\psi,\boldsymbol{p}) + \sigma J_{P,2}(\boldsymbol{p}), \tag{6-30}$$

with switch variable $\sigma = 0$ if $J_{P,1} > 0$ and $\sigma = 1$ otherwise. The second component activates once a suitable metric is found (i.e., once $J_{P,1}$=0). It aims to improve the estimation of $\boldsymbol{p}$ while looking for a better metric. Formally, it is defined as

$$J_{P,2}(\boldsymbol{p}) = \overbrace{w_5\ln(\epsilon^2+1)+w_6\ln(\rho^2+1)}^{\text{better synchro.}} - \underbrace{w_7\ln(\varepsilon^2+1)}_{\text{better Killing.}} - \overbrace{w_8\ln(\underline{p}^2+1)}^{\text{better Pos. Def.}}, \tag{6-31}$$

with $w_{P,2}$:=$(w_5,\ldots,w_8)$ is a vector of (positive) scalar weights. The composite objective (6-30) switches between metric search and contraction parameters optimization. First, it looks for a suitable metric along with a set of parameters $\boldsymbol{p}$. Then, it freezes the metric parameters and tries improving the contraction parameters $\boldsymbol{p}$. If the metric still satisfies $J_{P,1}$=0, another step is taken in the direction of $\boldsymbol{p}$ improvement. If not, it unfreezes the metric parameters and the loop starts again. Note that, by using $J_{P,1}$ as a discriminant, we can set the final weights of P_θ to be the last one verifying the contraction condition $J_{P,1}$=0, thus guaranteeing approximate output tracking (at least locally).

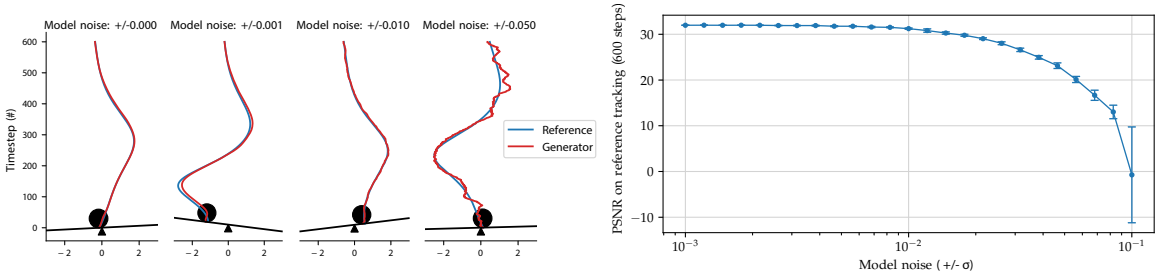
There are multiple advantages to using the switching objective (6-30). First, it achieves a better estimation of parameters $\boldsymbol{p}$. Second, it improves controller robustness, e.g., smaller ε implies faster contraction, that is, better stability margins (Sontag, 2010). Third, it weakens the dependence of $\boldsymbol{p}$ on the initial condition. As a matter of fact, $\boldsymbol{p}$ can be initialized to looser bounds, which eases training. The second objective will then try to tighten the conditions (6-28) progressively. Finally, it can escape from local minima as the shape of the loss function drastically changes on switches.

Step 3: Computing the stabilizer β

Once a suitable P_θ metric has been found, β_θ can be learned relatively to the following cost:

$$J_\beta(\boldsymbol{s},\psi) = \left|\frac{\partial\beta_\theta}{\partial\boldsymbol{s}}(\boldsymbol{s},\psi) - g(\boldsymbol{s})^\top P_\theta(\boldsymbol{s},\psi)\right|^2. \tag{6-32}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 6.2: **Qualitative evaluation of the steady state generator** – (a) We show four estimations from the state reference generator in different regimes where uniform noise is added to the model. (b) We measure the peak signal-to-noise ratio (PSNR, dB) between the reference and the output for different noise ranges. We show that our approach is robust up to a sensible amount of noise.

Each model is trained with Adam optimizer until convergence on a training set composed of states and commands $(\boldsymbol{s}, \psi)$ from the pre-trained state reference generator. Intermediate derivatives in (6-28) are obtained via automatic differentiation. The state reference generator and the stabilizer can be trained separately, as long as the training samples ψ_k for the stabilizer come from a similar distribution to the one of the output ψ of the state reference generator. Training P_θ, β_θ on the outputs of the state reference generator is a way to ensure this.

6.5 Simulations

We test our solution on the well-known ball and beam setup. The plant can be described by a system of the form (6-17) (Hauser et al., 1992) where $\boldsymbol{s} \in \mathbb{R}^4$, $u \in \mathbb{R}$ and

$$f(\boldsymbol{s}) = \begin{pmatrix} s_2 \\ B(s_1 s_4^2 - g_a \sin(s_3)) \\ s_4 \\ 0 \end{pmatrix}, \; g(\boldsymbol{s}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \; h(\boldsymbol{s}) = s_1 \,.$$

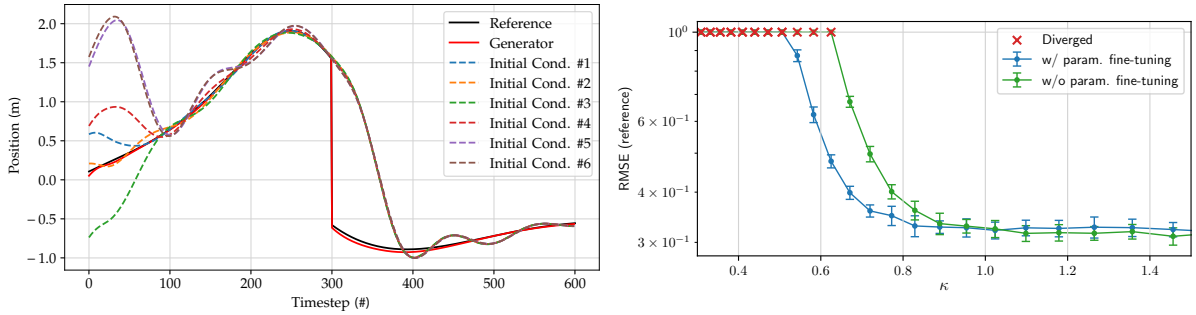
with B a constant depending on system parameters and g_a the gravitational acceleration. The interest of this setup lies in the fact that the relative degree[3] is not well-defined when the beam angular velocity and the ball position are zero. Therefore, input-output linearization and normal form-based approaches fail to give a suitable controller. To make the problem harder, we sample the reference signal using the trajectory of the first component z_1 of a Lorenz oscillator whose dynamics is described by

$$\begin{cases} \dot{z}_1 = 10(z_2 - z_1) \\ \dot{z}_2 = z_1(28 - z_3) - z_2 \\ \dot{z}_3 = z_1 z_2 - \frac{8}{3} z_3 \,, \end{cases} \tag{6-33}$$

with random initial conditions. As (6-33) is a chaotic oscillator, it is exponentially sensitive to initial conditions, making it hard to find analytical solutions to the regulator equations. Then,

[3] see (Isidori, 1995, Chapter IV)

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b)

Figure 6.3: **Output tracking** – (a) the stabilizer rapidly converges to the state reference even when the initial state is far from the generator at timestep k=0. At k=300 timesteps, we abruptly change the reference signal. The state reference generator and the stabilizer react accordingly. (b) Our fine-tuning step of the P metric increases the range of gain κ allowed for the stabilizer to track the reference. The red cross indicates that the system diverged from the reference signals.

Parameter Finetuning	Noise standard deviation			
	0	0.01	0.05	0.1
With	**0.343** $\pm$ 0.015	**0.323**$\pm$ 0.015	**0.361**$\pm$ 0.011	**0.385**$\pm$ 0.014
Without	0.385 $\pm$ 0.015	0.366 $\pm$ 0.016	0.408 $\pm$ 0.012	0.439 $\pm$ 0.016

Table 6.1: **Noise robustness** – is improved when using β_θ trained with the parameter fine-tuning step. We measure RMSE from reference for different standard deviation of Gaussian noise on state measurements. Our model can still perform correctly with uncertain observations.

approaches as in (Pavlov et al., 2006) become unfeasible in practice.

State reference generator – We demonstrate the quality of the estimated steady-state reference in figure 6.2a. Our approach succeed in providing a reliable pair $\left\{ \left(\psi[k], \pi[k]\right) \mid k \in [\![0, K]\!] \right\}$ solutions to the regulator equations. Note that the solution might not be unique, (multiple trajectories can give the same output), and we expect our model to provide an approximate solution. Figure 6.2b shows the evolution of the error when the dynamical model of the state reference generator is disturbed by a uniform noise whose amplitude is varied. The performance is measured on a set of new references absent from the training dataset. The standard deviation σ (represented by vertical bars) is obtained by averaging the results over five iterations. We find that the state reference generator is consistently robust to model errors up to a significant intensity.

Stabilizer – We generated the steady-state trajectory using the generator and used the simulator to control six instances of the system starting from different initial conditions. The result is illustrated in figure 6.3a. We find that each instance quickly converges towards the trajectory of the state reference generator. This is in accordance with the previous theoretical results. At timestep k=300, we drastically change the reference signal. The generator reacts immediately to such a change and estimates a new (π_k, ψ_k). Thanks to the stabilizer, the sys-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

tem converges quickly to the new reference. Once finished, the trajectory remains close to the state reference without deviating from it. As mentioned above, the analytical solution to the output tracking problem is difficult to obtain with such a nonlinear system under chaotic references. Although it is an approximation of the analytical solution, our approach experimentally demonstrates very satisfactory performance. We report quantitative results in Table 6.1, in particular in the context of a system perturbed by Gaussian noise modeling measurement errors. We observe that the learned stabilizer β_θ is robust even in noisy scenarios.

We also evaluate the advantages given by our switching objective, which reaches more stringent parameters $\boldsymbol{p}$. Table 6.1 compares noise robustness of our approach to the one without the fine-tuning component. We observe experimentally that our improved loss function leads to more robust control laws. Moreover, we also observe that fine-tuning allows for lower control gain κ, (Figure 6.3b). This is linked to the size of the domain of attraction when the Killing vector property holds only approximately (Giaccagli et al., 2022b).

6.6 Conclusions

Output tracking covers many applications, but proves itself to be very challenging, especially when the dynamics is nonlinear. The main interest of our method is that it can be applied readily to any nonlinear dynamics, provided that it is input-affine (e.g. most robotic systems). By grounding the structure of the controller into contraction theory, we obtained a robust and powerful solution that achieved good performance. Moreover, our algorithm does not involve online optimization and can be trained entirely offline, offering the potential requirements in terms of speed and memory footprint for on-board applications.

6.7 Post-scriptum: taking a step back

This work has been conducted in collaboration with two fellow PhD students experts in control theory and especially in contraction-based control. This is arguably the most transdisciplinary chapter of this manuscript, and this project required many discussions between both domains to translate the theoretical results from control to practical implementation using deep learning. We believe this work lays the foundation for promising future work. Solving the Riccati-like equation for nonlinear systems is tremendously difficult to achieve, especially with a Riemannian metric. Our method uses insights presented in part II, namely contraction and finite-time observability, but also harness recurrent models, auto-regressive forecasting, and PINNs, which play an important role in the next chapters.

Dynamical model of the system – We mentioned the recourse to a dynamical model of the system to train the steady-state generator (step 1). At first glance, this might appear as a strong limitation of our approach. Yet, in practice, it is uncommon to work on a plant without any model of its dynamics, and assuming the availability of a dynamical model is reasonable

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

in most cases. Moreover, our experiments in noisy scenarios indicate that our method can handle imprecise dynamical models. However, it is still possible to get rid of the model by relying either on reinforcement learning or on supervised learning.

Heavy machinery – We acknowledge that step 2 is convoluted, to say the least. In particular, learning the Riemannian metric involves back-propagating the gradients through the eigenvalue decomposition of four matrices M_i computed from not only the output of a neural network but also its derivative with regard to its inputs. Fortunately, this can be done easily with modern automatic differentiation tools, however, PINNs are known to be ill-posed problems and challenging to train, our situation is arguably on another level. There is room for simplification and improvement of the training pipeline, which is an interesting direction for future work.

Failure cases – Empirically, we observed good performance on a challenging setup. The input-affine structure of the dynamics is suitable for most robotics applications, thus offering a wide range of possible test systems. However, we discovered a failure case when testing our approach on the 2D drone system introduced in chapter 4. The method fails at step 2, and cannot identify a metric P satisfying all the conditions. We argue that drones are a particular case of dynamical systems which are extremely difficult to handle, especially in open-loop. Indeed, the dynamics of a drone is unstable: without inputs, the (numerically simulated) drone falls indefinitely. In practice, the UAV is operated in a very small area of the state space where it remains controllable. In other terms, the trajectories gathered from a drone for training purposes explore a very limited part of the dynamics, hence the difficulty in training a model on such a system. Actually, most applications of deep learning for drones work in closed-loop frameworks (Bauersfeld et al., 2021; Kaufmann et al., 2023; Loquercio et al., 2021).

Observer design – Another direct application of contraction theory and our method concerns observer design. Indeed, (Bernard et al., 2022, Theorem 4.1) states the existence of an observer for nonlinear autonomous systems $\dot{\boldsymbol{s}} = f(\boldsymbol{s}), y = h(\boldsymbol{s})$ of the form

$$\dot{\hat{\boldsymbol{s}}} = f(\hat{\boldsymbol{s}}, t) + k(\hat{\boldsymbol{s}}, y) \tag{6-34}$$

with

$$k(\hat{\boldsymbol{s}}, y) = \frac{1}{2}\kappa P(\hat{\boldsymbol{s}})\frac{\partial h}{\partial \boldsymbol{s}}^{\top}\big(y - h(\boldsymbol{s})\big) \tag{6-35}$$

and $P(\hat{\boldsymbol{s}})$ being a riemannian matrix solution of

$$L_f P + \rho(\boldsymbol{s}, t)\frac{\partial h}{\partial \boldsymbol{s}}^{\top}\frac{\partial h}{\partial \boldsymbol{s}} \preccurlyeq -qP \tag{6-36}$$

for $\rho : \mathbb{R}^{n_s} \times \mathbb{R}^{+} \mapsto \mathbb{R}$ and $q > 0$. This definition is very similar to ours, hence the method can be adapted. Interestingly, even if an exact solution over the entire state-space is not available, it may also be possible to still estimate approximate metric P avoiding the singularity points.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 7

Space and time continuous physics simulation from partial observations

Under review,
Co-authors: Madiha Nadri (Université Lyon 1),
Julie Digne (CNRS, Lyon),
Christian Wolf (NaverLabs Europe)

7.1 Context

In the previous chapter, we addressed a complex control algorithm using a Physics-Informed Neural Network (PINN) setup. Yet, PINNs have strong drawbacks when it comes to solving difficult PDEs. In this chapter, we propose a new data-driven solver for physics simulation which overcomes these limitations. In particular, we account for additional requirements on the behavior of the simulator:

R1. **Data-driven** – the underlying physics equation is assumed to be completely unknown. This includes the PDE, but also the boundary conditions. The dynamics must be discovered from a finite dataset of trajectories, i.e. a collection of observed behaviors from the physical system.
R2. **Generalization** – the method must generalize to new unseen initial conditions (ICs) readily, without re-training or fine-tuning.
R3. **Time and space continuous** – the domain of the predicted solution must be continuous in space and time[1] so that it can be queried at any arbitrary location within the domain of definition.

These requirements are common in the field but rarely addressed altogether. R1 allows for handling complex phenomena where the exact equation might be unknown, and R2 supports the growing need for faster simulators, which consequently must handle new ICs. Space and time continuity (R3) are also useful properties for standard simulations since the solution can

[1] In what follows while being a misnomer, *space and time continuity* of the solution designates the continuity of the spatial and temporal domain of definition of the solution, and not the continuity of the solution itself.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

be made as fine as needed in certain complex areas.

This task requires learning from sparsely distributed observations only and without any prior knowledge of the PDE form. In these settings, a standard approach consists of approximating the behavior of a discrete solver, enabling forecasting in an auto-regressive fashion (Pfaff et al., 2020; Yin et al., 2022; Janny et al., 2023; Sanchez-Gonzalez et al., 2020), losing therefore spatial and temporal continuity. Indeed, auto-regressive models assume strong regularities in the data, such as a static spatial lattice and uniform time steps. For these reasons, generalization to new spatial locations or intermediate time steps is not straightforward. These methods satisfy R1 and R2, but not R3. In another trend, PINNs learn a solution on a continuous domain. They leverage the PDE operator to optimize the weights of a neural network representing the solution, and cannot generalize to new ICs, thus violating R1 and R2.

We identified three state-of-the-art methods capable (at least in theory) of satisfying each requirement:

- **Neural operators** – can theoretically handle arbitrary locations and new initial conditions while not requiring the PDE operator. However, in practice, existing implementations are limited to discrete locations, or uniform sampling grid (Lu et al., 2019; Li et al., 2020d,c).
- **DINo** (Yin et al., 2022) – is a unique approach where the solution is modeled as a dynamically modulated implicit representation using a hyper-network. This method satisfies all three requirements and has been compared favorably to neural operators. However, it involves heavy machinery which may hinder its performance in complex scenarios, as shown empirically in this chapter.
- **Interpolate & Forecast** – is a different approach that leverages a discrete solver coupled to an interpolation module to provide time and space continuity. This method is illustrated in MaGNet (Boussif et al., 2022).

In this chapter, we address R1, R2 et R3 altogether in a new setup involving two joint dynamical systems. R1 and R2 are satisfied using auto-regressive discrete-time dynamics learned from the sparse observations and producing a trajectory in latent space. Then, R3 is achieved with a state observer derived from a second dynamical system in continuous time. This state observer relies on transformer-based cross-attention to enable evaluation at arbitrary spatio-temporal locations. In a nutshell: **(a)** We propose a new setup to address continuous space and time simulations of physical systems from sparse observation, leveraging insights from control theory. **(b)** We provide strong theoretical results indicating that our setup is well-suited to address this task compared to existing baselines, which are confirmed experimentally. **(c)** We provide experimental evidence that our state observer is more powerful than handcrafted interpolations for the targeted task. **(d)** With experiments on three challenging standard datasets (*Navier* Yin et al. (2022); Stokes (2009), *Shallow Water* Yin et al. (2022); Galewsky et al. (2004),

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

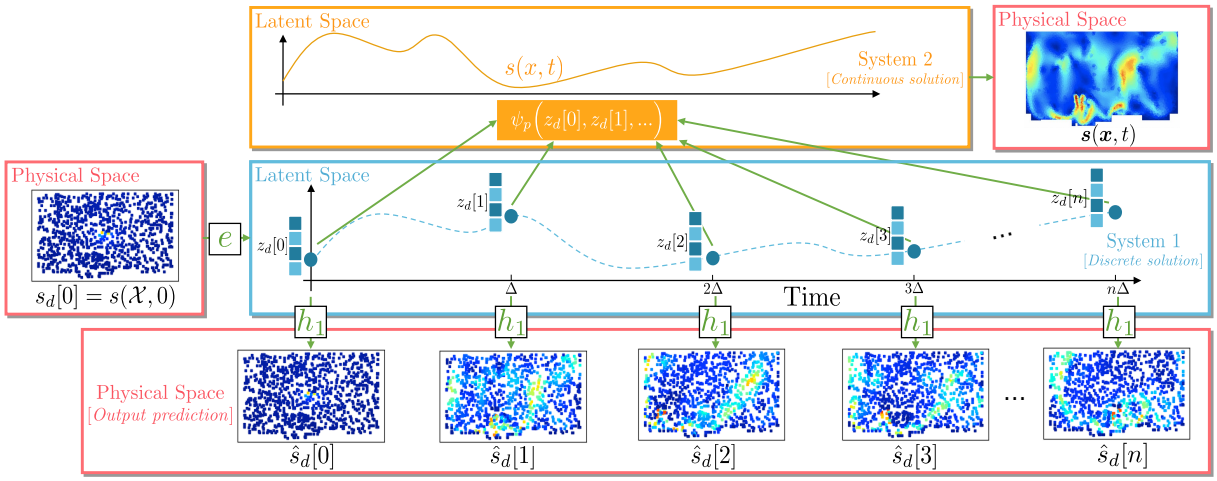


Figure 7.1: **Model overview** – We achieve space and time continuous simulations of physics systems by formulating the task as a double observation problem. System 1 is a discrete dynamical model used to compute a sequence of latent anchor states $\boldsymbol{z}_d$ auto-regressively, and System 2 is used to design a state estimator ψ_q retrieving the dense physical state at arbitrary locations $(\boldsymbol{x}, t)$.

Eagle Janny et al. (2023), and against state-of-the-art methods (MeshGraphNet (MGN) Pfaff et al. (2020), DINO Yin et al. (2022), MAgNet (Boussif et al., 2022)).

7.2 Continuous solutions from sparse observations

Consider a dynamical system following a PDE defined for all $(\boldsymbol{x}, t) \in \Omega \times [\![0, T]\!]$, with T a positive constant:

$$\begin{aligned} \dot{\boldsymbol{s}}(\boldsymbol{x}, t) &= f\left(\boldsymbol{s}, \boldsymbol{x}, t\right) \quad \forall (\boldsymbol{x}, t) \in \Omega \times [\![0, T]\!], \\ \boldsymbol{s}(\boldsymbol{x}, 0) &= \boldsymbol{s}_0(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega, \quad \boldsymbol{s}(\boldsymbol{x}, t) = \bar{\boldsymbol{s}}(\boldsymbol{x}, t) \quad \forall (\boldsymbol{x}, t) \in \partial\Omega \times [\![0, T]\!] \end{aligned} \tag{7-1}$$

where the state lies in an invariant set $\boldsymbol{s} \in \mathcal{S}$, $f : \mathcal{S} \mapsto \mathcal{S}$ is an unknown operator, $\boldsymbol{s}_0 : \Omega \mapsto \mathbb{R}^n$ is the initial condition (IC) and $\bar{\boldsymbol{s}} : \partial\Omega \times [\![0, T]\!] \mapsto \mathbb{R}^n$ the boundary condition. In what follows, we consider trajectories with shared boundary conditions, hence we omit $\bar{\boldsymbol{s}}$ from the notation for readability. In practice, the operator f is unknown, and we assume access to a set $\mathcal{D}$ of K discrete trajectories from different ICs $\boldsymbol{s}_0^k$, sampled at sparse and scattered locations in time and space. Formally, we introduce two finite sets $\mathcal{X} \subset \Omega$ of fixed positions and fixed regularly sampled times $\mathcal{T}$ at sampling rate Δ^*. Let $S(\boldsymbol{s}_0, \boldsymbol{x}, t)$ be the solution of this PDE from IC $\boldsymbol{s}_0$, the dataset $\mathcal{D}$ is given as: $\mathcal{D} := \left\{ S(\boldsymbol{s}_0^k, \mathcal{X}, \mathcal{T}) \;\middle|\; k \in [\![1, K]\!] \right\}$. Our task is formulated as:

Given $\mathcal{D}$, a new initial condition $\boldsymbol{s}_0 \in \mathcal{S}$, and a query $(\boldsymbol{x}, t) \in \Omega \times [\![0, T]\!]$, find the solution of equation(7-1) at the queried location and from the given IC, that is $S(\boldsymbol{s}_0, \boldsymbol{x}, t)$.

Note that this task involves generalization to new ICs, as well as estimation to unseen spatial locations within Ω and unseen time instants within $[\![0, T]\!]$. We do not explicitly require extrapolation to instants $t > T$, although it comes as a side benefit of our approach up to some extent.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

7.2.1 *The double observation problem*

The task implies extracting regularities from weakly informative physical variables that are sparsely measured in space and time since $\mathcal{X}$ and $\mathcal{T}$ contain very few elements. Consequently, the possibility to forecast their trajectories from off-the-shelf auto-regressive methods is very unlikely (as confirmed experimentally). To tackle this challenge, we propose an approach accounting for the fact that the phenomenon is not directly observable from the sparse trajectories, but can be deduced from a richer latent state-space in which the dynamics is Markovian. We introduce two linked dynamical models lifting sparse observations to dense trajectories guided by *observability* considerations, namely

$$\underline{\text{System 1:}} \begin{cases} \boldsymbol{z}_d[n+1] &= f_1\big(\boldsymbol{z}_d[n]\big) \\ \boldsymbol{s}_d[n] &= h_1\big(\boldsymbol{z}_d[n]\big) \end{cases}, \quad \underline{\text{System 2:}} \begin{cases} \dot{\boldsymbol{s}}(\boldsymbol{x},t) &= f_2\big(\boldsymbol{s},\boldsymbol{x},t\big) \\ \boldsymbol{z}(\boldsymbol{x},t) &= h_2\big(\boldsymbol{s},\boldsymbol{x},t\big) \end{cases} \quad \forall (\boldsymbol{x},t)\in\Omega\times[\![0,T]\!] \tag{7-2}$$

where for all $n \in \mathbb{N}$, we note $\boldsymbol{s}_d[n] = \boldsymbol{s}(\mathcal{X}, n\Delta)$ the sparse observation at some instant $n\Delta$. The sampling rate Δ is not necessarily equal to the sampling rate Δ^* used for data acquisition, which we will exploit during training to improve generalization. This will be detailed later.

System 1 – is a discrete-time dynamical system where the available measurements $\boldsymbol{s}_d[n]$ are considered as *partial observations* of a latent state variable $\boldsymbol{z}_d[n]$. We aim to derive an output predictor from System 1 to forecast trajectories of sparse observations auto-regressively from the sparse IC. As mentioned earlier, sparse observations are unlikely to be sufficient to perform predictions, hence we introduce a richer latent state variable $\boldsymbol{z}_d$ in which the dynamics is truly Markovian, and observations $\boldsymbol{s}_d[n]$ are seen as measurements of the state $\boldsymbol{z}_d$ using the function h_1.

System 2 – is a continuous-time dynamical system describing the evolution of the to-be-predicted dense trajectory $S(\boldsymbol{s}_0, \boldsymbol{x}, t)$. It introduces continuous observations $\boldsymbol{z}(\boldsymbol{x}, t)$ such that $\boldsymbol{z}(\mathcal{X}, n\Delta) = \boldsymbol{z}_d[n]$. The insight is that the state representation $\boldsymbol{z}_d[n]$ obtained from System 1 is designed to contain sufficient information to predict $s_d[n]$, but not necessarily to predict the dense state. Formally, $\boldsymbol{z}_d$ represents solely the *observable part* of the state, in the sense of control theory.

At inference time, we forecast at query location $(\boldsymbol{x}, t)$ with a 2-step algorithm: **(Step-1)** System 1 is used as an output predictor from the sparse IC $\boldsymbol{s}_d[0]$, and computes a sequence $\boldsymbol{z}[0], \boldsymbol{z}[1], \cdots$, which we refer to as "*anchor states*". This sequence allows the dynamics to be Markovian, provides sufficient information for the second state estimation step, and holds information to predict the sparse observations, allowing supervision during training. **(Step-2)** We derive a *state observer* from System 2 leveraging the anchor states over the whole time domain to estimate the dense solution at an arbitrary location in space and time (see figure 7.1). Importantly, for a given IC, the anchor states are computed only once and reused within System 2 to estimate the solution at different points.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

7.2.2 *Theoretical analysis*

In this section, we introduce theoretical results supporting the use of Systems 1 and 2. In particular, we show that using System 1 to forecast the sparse observations in latent space $\boldsymbol{z}_d$ rather than directly operating in the physical space leads to a smaller upper bound on the prediction error. Then, we show the existence of a state estimator from System 2 and compute an upper bound on the estimation error depending on the length of the sequence of anchor states.

Step 1 – consists of computing the sequence of anchor states guided by an output prediction task of the sparse observations. As classically done, we introduce an encoder (formally, a state observer) $e(\boldsymbol{s}_d[0]) = \boldsymbol{z}_d[0]$ coupled to System 1 to project the sparse IC into a latent space $\boldsymbol{z}_d$. Following System 1, we compute the anchor states $\boldsymbol{z}_d$ auto-regressively (with f_1) in the latent space. The sparse observations are extracted from $\boldsymbol{z}_d$ using h_1. In comparison, existing baselines (Pfaff et al., 2020; Sanchez-Gonzalez et al., 2020; Stachenfeld et al., 2021) maintain the state in the physical space and discard the intermediate latent representation between iterations. Formally, let us consider approximations $\hat{f}_1, \hat{h}_1, \hat{e}$ (in practice realized as deep networks trained from data $\mathcal{D}$) of f_1, h_1 and e and compare the prediction algorithm for the classic auto-regressive (AR) approach and ours

$$\underline{\text{Classic AR:}} \quad \hat{\boldsymbol{s}}_d^{\text{ar}}[n] := (\hat{h}_1 \circ \hat{f}_1 \circ \hat{e})^n(\boldsymbol{s}_d[0]) \qquad \underline{\text{Ours:}} \quad \hat{\boldsymbol{s}}_d[n] := \hat{h}_1 \circ \hat{f}_1{}^n \circ \hat{e}(\boldsymbol{s}_d[0]) \tag{7-3}$$

Classical AR approaches re-project the latent state into the physical space at each time step and repeat the scheme "encode-process-decode". Our method encodes the sparse IC, advances the system in the latent space, and decodes toward the physical space at the end. The following proposition indicates that our method is advantageous compared to the classic AR scheme.

Proposition 7.1 *Consider a dynamical system of the form of System 1 and assume the existence of a state observer e along with approximations $\hat{f}_1, \hat{h}_1, \hat{e}$ with Lipschitz constants L_f, L_h and L_e respectively such that $L_h L_f L_e \neq 1$. If there exist $\delta_f, \delta_h, \delta_e \in \mathbb{R}^+$ such that $\forall (\boldsymbol{z}, \boldsymbol{s}) \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_s}$*

$$|f_1(\boldsymbol{z}) - \hat{f}_1(\boldsymbol{z})| \leqslant \delta_f, \quad |h_1(\boldsymbol{z}) - \hat{h}_1(\boldsymbol{z})| \leqslant \delta_h, \quad |e(\boldsymbol{s}) - \hat{e}(\boldsymbol{s})| \leqslant \delta_e \tag{7-4}$$

for the Euclidean norm $|\cdot|$, then for all integer $n > 0$, with $\hat{\boldsymbol{s}}_d[n]$ and $\hat{\boldsymbol{s}}_d^{ar}[n]$ as in (7-3),

$$|\boldsymbol{s}_d[n] - \hat{\boldsymbol{s}}_d[n]| \leqslant \delta_h + L_h \left(\delta_f \frac{L_f^n - 1}{L_f - 1} + L_f^n \delta_e \right) \tag{7-5}$$

$$|\boldsymbol{s}_d[n] - \hat{\boldsymbol{s}}_d^{ar}[n]| \leqslant \delta \frac{L^n - 1}{L - 1} \tag{7-6}$$

with $\delta = \delta_h + L_h \delta_f + L_h L_f \delta_e$ and $L = L_h L_f L_e$.

Proof: See appendix D.1.

This result shows that falling back to the physical space at each time step degrades the upper bound of the prediction error. Indeed, if $L < 1$, the upper bound converges trivially to

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

zero when n increases, and hence can be ignored. Otherwise, the upper bound for the classic AR scheme appears to be more sensitive to approximation errors δ_h, δ_f and δ_e compared to our approach (for a formal comparison, see appendix D.2). Intuitively it means that information is lost in the observation space, which thus needs to be re-estimated at each iteration when using the classic AR scheme. By maintaining a state variable in the latent space, we allow this information to flow readily between each step of the simulator (see blue frame in figure 7.1).

Step 2 – The state estimator builds upon System 2 and relies on the set of anchor states from the previous step to estimate the dense physical state at arbitrary locations in space and time. Formally, we look for a function ψ_q leveraging the sequence of anchor states $\boldsymbol{z}_d[0], \cdots, \boldsymbol{z}_d[q]$ (simulated from the sparse IC $\boldsymbol{s}_d[0]$) to retrieve the dense solution[2]. In what follows, we show that (1) such a function ψ_q exists and (2) we compute an upper bound on the estimation error depending on the length of the sequence. To do so, consider the functional which outputs the anchor states from any IC $\boldsymbol{s}_0 \in \mathcal{S}$

$$\mathcal{O}_p(\boldsymbol{s}_0) = \Big[\; h_2(\boldsymbol{s}_0(\mathcal{X}))\; h_2(S(\boldsymbol{s}_0, \mathcal{X}, \Delta))\; \cdots\; h_2(S(\boldsymbol{s}_0, \mathcal{X}, p\Delta))\;\Big] = \Big[\; \boldsymbol{z}_d[0]\; \cdots\; \boldsymbol{z}_d[p]\;\Big] \tag{7-7}$$

In practice, the ground truths $\boldsymbol{z}_d[n]$ are not perfectly known, as they are obtained from a data-driven output predictor (step 1) using the sparse IC. Inspired from chapter 4, we state:

Proposition 7.2 *Consider a dynamical system defined by System 2 and (7-7). Assume that*

A1. *f_2 is Lipschitz with constant L_s,*

A2. *there exists $p > 0$ and a strictly increasing function α such that $\forall \boldsymbol{s}_a, \boldsymbol{s}_b \in \mathcal{S}^2$ and $\forall q \geqslant p$*

$$\big|\mathcal{O}_q(\boldsymbol{s}_a) - \mathcal{O}_q(\boldsymbol{s}_b)\big| \geqslant \alpha(q)|\boldsymbol{s}_a - \boldsymbol{s}_b|_{\mathcal{S}} \tag{7-8}$$

where $|\cdot|_{\mathcal{S}}$ is an appropriate norm for $\mathcal{S}$.

Then, $\forall q \geqslant p$, there exists ψ_q such that, for $(\boldsymbol{x}, t) \in \Omega \times [\![0, T]\!]$ and δ_n such that $\hat{\boldsymbol{z}}_d[n] = \boldsymbol{z}_d[n] + \delta_n$, for all $n \leqslant q$,

$$\psi_q\big(\boldsymbol{z}_d[0], \cdots, \boldsymbol{z}_d[q], \boldsymbol{x}, t\big) = S(\boldsymbol{s}_0, \boldsymbol{x}, t) \tag{7-9}$$

$$\Big|S(\boldsymbol{s}_0, \boldsymbol{x}, t) - \psi_q\big(\hat{\boldsymbol{z}}_d[0], \cdots, \boldsymbol{z}_d[q], \boldsymbol{x}, t\big)\Big|_{\mathcal{S}} \leqslant 2\alpha(q)^{-1}\big|\delta_{0|q}\big|e^{L_s t}. \tag{7-10}$$

where $\delta_{0|q} = \big[\delta_0\; \cdots\; \delta_q\big]$.

Proof: See appendix D.3.

Assumption A2. states that the longer we observe two trajectories from different ICs, the easier it will be to distinguish them, ruling out systems collapsing to the same state. Such systems are uncommon since forecasting their trajectory becomes trivial after some time. This assumption is related to *finite-horizon observability* in control theory, a property of dynamical systems guaranteeing that the (markovian) state can be retrieved given a finite

[2] Since the simulation is conducted up to T, and considering the time step Δ, in practice $q \leqslant \lfloor \frac{T}{\Delta} \rfloor$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

number p of past observations. Equation 7-8 is associated with injectivity of $\mathcal{O}_q$, hence the existence of a left inverse mapping the sequence of anchor states to the IC $\boldsymbol{s}_0$.

Proposition 7.2 highlights a trade-off on the performance of ψ_q. On one hand, longer sequences of anchor states are harder to predict, leading to a larger $|\delta_{0|q}|$, which impacts the state estimator ψ_q negatively. On the other hand, longer sequences hold more information that can still be leveraged by ψ_q to improve its estimation, represented by $\alpha(q)^{-1}$ in (7-10). In contrast to competing baselines or conventional interpolation algorithms, our approach takes this trade-off into account by explicitly leveraging the sequence to estimate the dense solution, as will be discussed below.

Discussion and related work – the competing baselines can be analyzed using our setup, yet in a weaker configuration. For instance, one can see Step 2 as an interpolation process, and replace it with a conventional interpolation algorithm, which typically relies on spatial neighbors only. Our method not only exploits spatial neighborhoods but also leverages temporal data, improving the performance, as shown in proposition 7.2 and empirically corroborated in Section 7.3.

MAgNet (Boussif et al., 2022) uses a reversed *interpolate-forecast* scheme compared to ours. The IC $\boldsymbol{s}_d[0]$ is interpolated right from the start to estimate $\boldsymbol{s}_0$ (corresponding to our Step 2, with q=1), and then simulated with an auto-regressive model in the physical space (with the classic AR scheme). Propositions 7.1 and 7.2 show that the upper bounds on the estimation and prediction error are higher than ours. Moreover, if the number of query points exceeds the number of known points ($|\Omega| \gg |\mathcal{X}|$), the input of the auto-regressive solver is filled with noisy interpolations, which impacts performance. This is illustrated in figure D.1 in the appendix.

DINo (Yin et al., 2022) is a very different approach leveraging a spatial implicit neural representation modulated by a context vector, whose dynamics is modeled via a learned ODE. This approach is radically different than ours and arguably involves stronger hypotheses, such as the existence of a learnable ODE modeling the dynamics of a suitable weight modulation vector. In contrast, our method relies on strong insights from dynamical systems and observation theory.

7.2.3 *Implementation*

The implementation follows the algorithm described in the previous section: **(Step-1)** rolls out predictions of anchor states from the IC and **(Step-2)** estimates the state at the query position from these anchor states. The encoder $\hat{e}$ from Step 1 is an MLP which takes as input the sparse IC $\boldsymbol{s}_d[0]$ and the positions $\mathcal{X}$ and outputs a latent state variable $\boldsymbol{z}_d[0]$ structured as a graph, with edges computed with a Delaunay triangulation. Hence, each anchor is a graph $\boldsymbol{z}_d[n] = \{\boldsymbol{z}_d[n]_i\}$, but we will omit index i over graph nodes in what follows if not required for understanding.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

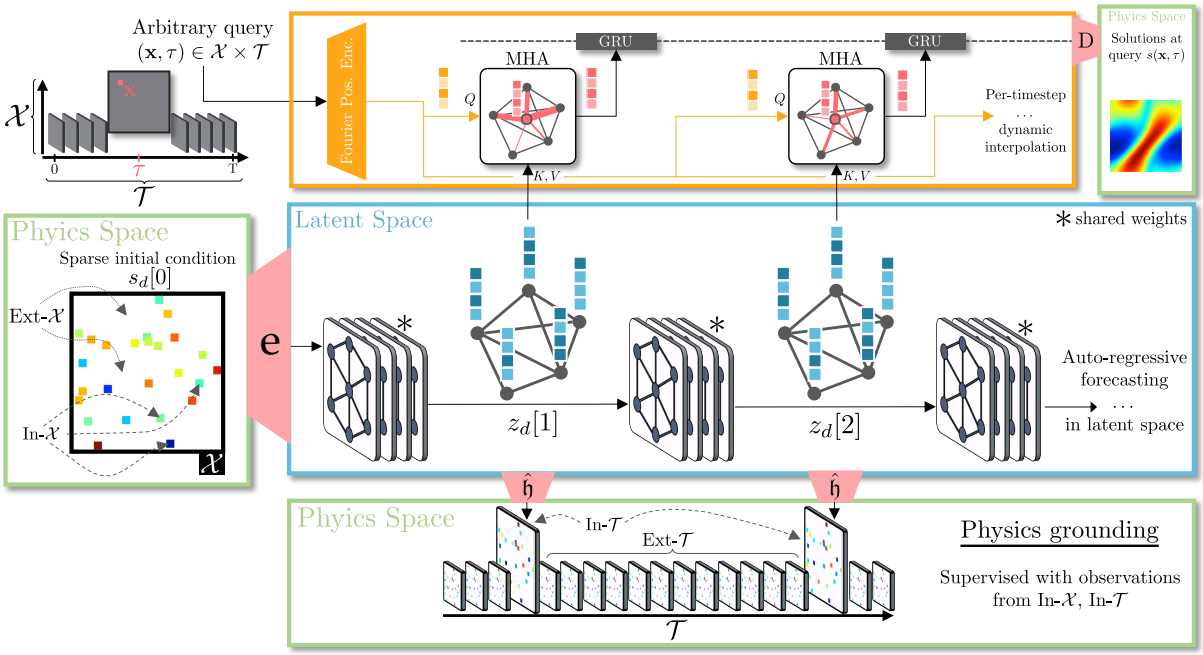


Figure 7.2: **Model overview** – The model leverages a dynamical system (System 1) to perform auto-regressive predictions of the dynamics in a mesh-structured latent space from sparse initial conditions. It is combined with a data-driven state estimator derived from another continuous-time dynamical system (System 2), implemented with multi-head cross-attention. The attention mechanism queries the intermediate anchor states from the auto-regressive predictor and uses Fourier positional encoding to encode the query points $(\mathbf{x}, \tau)$. An additional GRU refines the dynamics after interpolation.

We model $\hat{f}_1$ as a multi-layer GNN (Battaglia et al., 2016). The anchor states $z_d[n]$ are defined at fixed time steps $n\Delta$, which might not match Δ^* used in the data $\mathcal{T}$. We found it beneficial to choose $\Delta{=}k{\times}\Delta^*$ with $k{>}1{\in}\mathbb{N}$ such that the model can be queried *during training* on time points $t \in \mathcal{T}$ that do not match exactly with every time-steps in $z_d[0], z_d[1], ...$, hence encouraging generalization to unseen time. The observation function $\hat{h}_1$ is an MLP applied on the vector at node level in the graph z_d.

The state estimator ψ_q is decomposed into a Transformer model (Vaswani et al., 2017) coupled to a recurrent neural network to provide an estimate at spatio-temporal query position $(\boldsymbol{x}, t)$. First, through cross-attention we translate the set of anchor states $z_d[n]$ (one embedding per graph node i and per instant n) into a set of estimates of the continuous variable $\boldsymbol{z}(\boldsymbol{x}, t)$ *conditioned* at the instant $n\Delta$, which we denote $z_{n\Delta}(\boldsymbol{x}, t)$ (one embedding per instant n). Following advances in geometric mappings in computer vision (Saha et al., 2022), we use multi-head cross-attention to *query* from coordinates $(\boldsymbol{x}, t)$ the *Keys* corresponding to the nodes i in each graph anchor state $z_d[n]$, $\forall n$

$$z_{n\Delta}(\boldsymbol{x}, t) = f_{\text{mha}}\big(\text{Q}{=}\zeta_\omega(\boldsymbol{x}, t), \text{K}{=}\text{V}{=}\{z_d[n]_i\} + \zeta_\omega(\mathcal{X}, n\Delta)\big), \texttt{ // attention over nodes i} \tag{7-11}$$

where Q, K, V are, respectively, *Query*, *Key* and *Value* inputs to the cross-attention layer f_{mha} (Vaswani et al., 2017) and ζ_ω a Fourier positional encoding with a learned frequency parameter

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

ω. Finally, we leverage a state observer to estimate the dense solution at the query point from the sequence of conditioned anchor variables, over time. This is achieved with a GRU (Cho et al., 2014) maintaining a hidden state $\boldsymbol{u}[n]$,

$$\boldsymbol{u}[n] = r_{\text{gru}}\big(\boldsymbol{u}[n-1], \boldsymbol{z}_{n\Delta}(\boldsymbol{x}, t)\big), \quad \hat{S}(\boldsymbol{s}_0, \boldsymbol{x}, t) = D\left(\boldsymbol{u}[q]\right), \tag{7-12}$$

which shares similarities with conventional state-observer designs in control theory (Bernard et al., 2022). Finally, an MLP D maps the final GRU hidden state to the desired output, that is, the value of the solution at the desired spatio-temporal coordinate $(\boldsymbol{x}, t)$. See appendix D.4 and figure 7.2 details.

7.2.4 *Training*

Generalization to new input locations during training is promoted by creating artificial generalization situations using sub-sampling techniques of the sparse sets $\mathcal{X}$ and $\mathcal{T}$.

Artificial generalization – The anchor states $z_d[n]$ are computed at time rate Δ larger than the available rate Δ^*. This creates situations *during training* where the state estimator ψ_q does not have access to a latent state perfectly matching with the queried time. We propose a similar trick to promote spatial generalization. At each iteration, we sub-sample the (already sparse) IC $\boldsymbol{s}_d[0]$ randomly to obtain $\tilde{\boldsymbol{s}}_d[0]$ defined on a subset of $\mathcal{X}$. We then compute the anchor states $\tilde{\boldsymbol{z}}_d$ using System 1. On the other hand, the query points are selected in the larger set $\mathcal{X}$. Consequently, System 2 is exposed to positions that do not always match with the ones in $\boldsymbol{z}_d[n]$. Note that the complete domain of definition $\Omega \times [\![0, T]\!]$ remains unseen during training.

Training objective – To reduce training time, we randomly sample M query points $(\boldsymbol{x}_m, \tau_m)$ in $\mathcal{X} \times \mathcal{T}$ at each iteration, with a probability proportional to the previous error of the model at this point since its last selection (see appendix D.4) and we minimize the loss

$$\mathcal{L} = \overbrace{\sum_{k=1}^{K} \sum_{m=1}^{M} \left| S(\boldsymbol{s}_0^k, \boldsymbol{x}_m, \tau_m) - \psi_q\big(\tilde{\boldsymbol{z}}_d[0|q], \boldsymbol{x}, \tau_m\big) \right|^2}^{\mathcal{L}_{\text{continuous}}} + \overbrace{\sum_{n=0}^{\lfloor T/\Delta \rfloor} \left| \tilde{\boldsymbol{s}}_d[n] - \hat{h_1}\big(\tilde{\boldsymbol{z}}_d[n]\big) \right|^2}^{\mathcal{L}_{\text{dynamics}}}, \tag{7-13}$$

with $\tilde{\boldsymbol{z}}_d[n] = \hat{f_1}^n \circ \hat{e}\big(\tilde{\boldsymbol{s}}_d[0]\big)$. $\mathcal{L}_{\text{continuous}}$ supervises the model end-to-end, and $\mathcal{L}_{\text{dynamics}}$ trains the latent anchor states $\boldsymbol{z}_d$ to predict the sparse observations from the IC.

7.3 Experimental results

Experimental setup – we generate $\mathcal{X} \times \mathcal{T}$ from $\Omega \times [\![0, T]\!]$ by subsampling the domain of definition with various sampling rates, which changes the difficulty level of the task. We evaluate on three highly challenging datasets (details are provided in appendix D.5):

- **Navier** (Yin et al., 2022; Stokes, 2009) simulates the vorticity of a viscous, incompressible flow using the Navier-Stokes equations. The flow is driven by a sinusoidal force acting on a square domain with periodic boundary conditions.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

- **Shallow Water** (Yin et al., 2022; Galewsky et al., 2004) is a dataset defined on a non-euclidean manifold. It studies the velocity of shallow waters evolving on the tangent surface of a 3D sphere.
- **Eagle** (Janny et al., 2023) is a challenging dataset of unsteady airflow (turbulences) generated by a moving drone in a 2D environment with many different scene geometries. This dataset will be presented in depth in Chapter 8.

We also evaluate our model against three competitive baselines that represent the current state-of-the-art in continuous forecasting of physics solutions.

- **Interpolated MeshGraphNet (MGN)** (Pfaff et al., 2020) is a standard multi-layered GNN used in an auto-regressive manner. This baseline is extended to time and space continuity using physics-agnostic spline interpolation techniques.
- **MAgNet** (Boussif et al., 2022) proposes to interpolate the mesh to new positions in the latent space before applying an MGN model (complemented with an Euler time-discretization scheme to allow for arbitrary time querying). While it is designed to generalize to new locations in space, it also assumes knowledge of the evaluation graph during training, including new query points. When evaluating the super-resolution setup, the authors kept the ratio between the amount of new query points and available measurements constant. Hence, while the model is queried at unseen locations, it also benefits from more information. In our setup, the model is exposed to a fixed set of points and does not receive more samples when evaluated on positions unseen during training. This makes our task more challenging than the one originally addressed in Boussif et al. (2022).
- **DINo** (Yin et al., 2022) models the solution as an INR $\boldsymbol{s}(\boldsymbol{x}, \alpha_t)$ where the spatial coordinates $\boldsymbol{x}$ are fed to a *Multiplicative Filter Network* (MFN) (Fathony et al., 2021) and α_t is a context vector modulating the activations of the INR. The dynamics of the context vector is modeled with a Neural-ODE $\dot{\alpha}_t = F(\alpha_t)$, where F is an MLP. We share common objectives with DINo and take inspiration from the evaluation tasks in Yin et al. (2022), yet in a more challenging setup.

Note that all three baselines have been extensively described in chapter 3. Details of the baselines are in appendix D.5. We highlight a caveat on MAgNet: the model can handle a limited amount of new queries, roughly equal to the number of observed points. Our task requires the solution at up to 20 times more queries than available points. In this situation, the graph in MaGNet is dominated by noisy states from interpolation, and the auto-regressive forecaster performs poorly. During the evaluation, we found it beneficial to split the queries into chunks of 10 nodes and to apply the model several times. This strongly improves the performance at the cost of an increased runtime.

Space Continuity – Table 7.1 compares the spatial interpolation power of our method versus

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

		Navier			Shallow Water			Eagle	
		High	Mid	Low	High	Mid	Low	High	Low
DINo (Yin et al., 2022)	In-$\mathcal{X}$	1.557	1.130	1.878	**0.1750**	**0.1814**	0.2733	287.3	302.7
	Ext-$\mathcal{X}$	1.600	1.253	5.493	4.638	13.40	21.55	381.7	489.6
Interp. MGN (Pfaff et al., 2020)	In-$\mathcal{X}$	1.913	0.9969	0.6012	0.3663	0.2835	0.7309	**64.44**	83.58
	Ext-$\mathcal{X}$	2.694	4.784	14.80	1.744	4.221	8.187	173.4	241.5
Time Oracle (n.c)	In-$\mathcal{X}$		*n/a*			n/a		*n/a*	
	Ext-$\mathcal{X}$	*0.851*	*4.204*	*15.63*	*1.617*	*4.327*	*8.522*	*147.0*	*221.2*
MAgNet (Boussif et al., 2022)	In-$\mathcal{X}$	18.17	6.047	8.679	0.3196	0.3358	0.4292	99.79	124.5
	Ext-$\mathcal{X}$	35.73	26.24	57.21	10.21	23.20	30.55	194.3	260.7
Ours	In-$\mathcal{X}$	**0.1989**	**0.2136**	**0.2446**	0.2940	0.3139	**0.2700**	70.02	**78.83**
	Ext-$\mathcal{X}$	**0.2029**	**0.2463**	**0.5601**	**0.4493**	**1.051**	**2.800**	**90.88**	**117.2**

Table 7.1: **Space Continuity** – we evaluate the spatial interpolation power of our method vs. the baselines and standard interpolation techniques. We vary the number of available measurement points in the data for training from High (25% of simulation grid), Middle (10%), and Low (5%) amount of points and show that our model outperforms the baselines. Evaluation is conducted over 20 frames in the future (10 for *Eagle*) and we report the MSE to the ground truth solution ($\times 10^{-3}$).

several baselines. The MSE values computed on the training domain (In-$\mathcal{X}$=$\mathcal{X}$) and outside (Ext-$\mathcal{X}$=$\Omega \setminus \mathcal{X}$) show that our method offers the best performance, especially for the Ext-domain task, which is our aim. To ablate dynamics and evaluate the impact of trained interpolations, we also report the predictions of a *Time Oracle* which uses sparse ground truth values at all time steps and interpolates (bicubic) spatially. This allows us to assess whether the method is doing better than a simple axiomatic interpolation. While MGN offers competitive in-domain predictions, the cubic interpolation fails to extrapolate reliably on unseen points. This can be seen in the In/Ext gap for Interpolated MGN which is very close to the Time Oracle error. MaGNet, which builds on a similar framework, is hindered by the larger amount of unobserved data in the input mesh. At test time, the same number of initial condition points are provided but the method interpolates substantially more points. DINo achieves a very low In/Ext gap, yet fails on highly (5%) down-sampled tasks. One of the key difference with DINo is that the dynamics relies on an internal ODE for the temporal evolution of a modulation vector. In contrast, our model uses an explicit auto-regressive backbone and time forecasting is handled in an arguably more meaningful space, which we conjecture to be the reason why we achieve better results (see figure D.2 in the appendix for qualitative results on Shallow Water).

Time Continuity – is a step forward in difficulty, as the model needs to interpolate not only to unseen spatial locations (datasets are undersampled at 25%) but also on intermediate timesteps (Ext-$\mathcal{T}$, Table 7.2). All models perform well on *Shallow Water*, which is relatively easy. Both DINo and MAgNet leverage a discrete integration scheme (Euler for MAgNet and RK4 for DINo) allowing querying the model between timesteps seen at training. These schemes struggle to capture the data dependencies effectively and therefore the methods fail

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

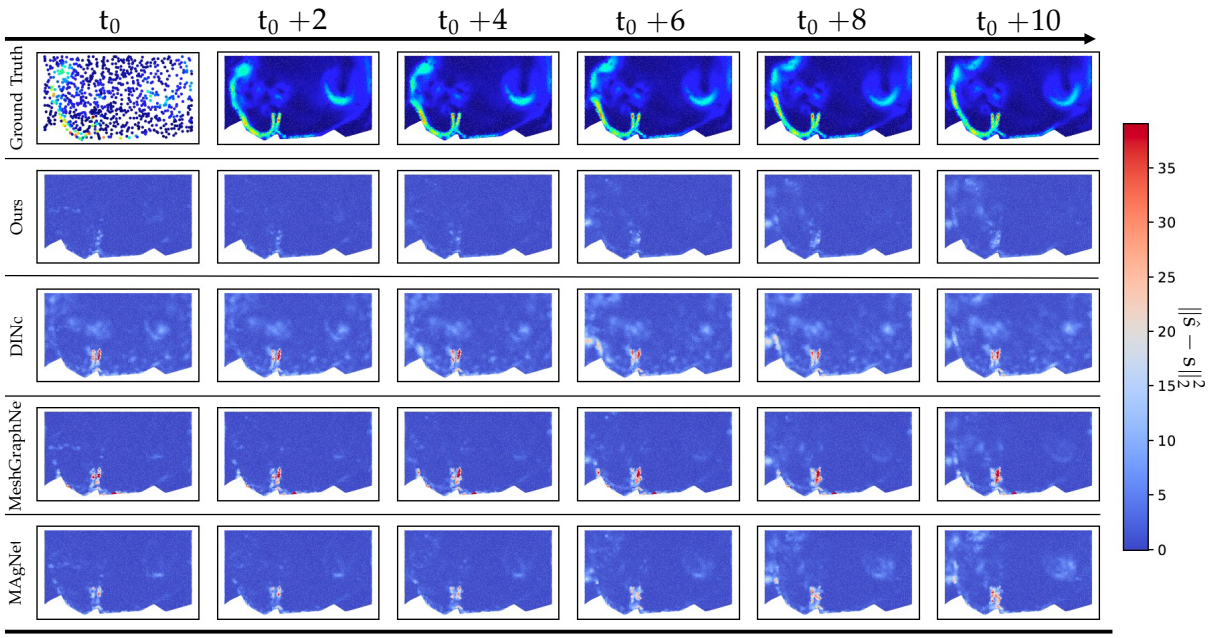
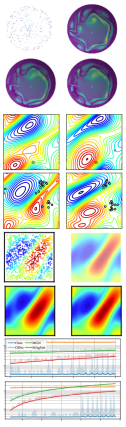


Figure 7.3: **Qualitative results on Eagle** – Per point error of the flow prediction on an *Eagle* example in the *Low* spatial down-sampling scenario. Our model shows lower errors, as also shown in Tables 7.1 and 7.2.

on *Navier* (see also Figure D.3 for qualitative results). *Eagle* is particularly challenging, the main source of error being the spatial interpolation, as can be seen in Figure 7.3 – our method yields lower errors in flow estimation.



Many more experiments – are available in appendix D.6. We study the **impact of key design choices**, artificial generalization, and dynamical loss. We show qualitative results on time interpolation and time extrapolation on the *Navier* dataset. We **explore generalization to different grids**. We provide more empirical evidence of the soundness of Step 2 in an **ablation study** (including comparison with attentive neural process (Kim et al., 2018), an attention-based structure somehow close to ours), and **observe attention maps** on several examples. We show that our **state estimator goes beyond local interpolation**, as conventional interpolation algorithms would do. Finally, we also measure the **computational burden** of the discussed methods and show that our approach is more efficient.

7.4 Conclusion

We used a double dynamical system formulation for simulating physical phenomena at arbitrary locations in time and space. Our approach comes with theoretical guarantees on existence and accuracy without knowledge of the underlying PDE. Furthermore, our method generalizes to unseen initial conditions and reaches excellent performance outperforming existing methods. Yet a limitation lies in the requirement of regular sampling in time, on which our dynamics loss can be expressed. Finally, in the case of known and well-studied phenom-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

		Navier			Shallow Water			Eagle		
		1/1	1/2	1/4	1/1	1/2	1/4	1/1	1/2	1/4
DINo (Yin et al., 2022)	In-$\mathcal{T}$	1.590	36.31	46.02	3.551	6.005	6.249	444.5	447.1	448.6
	Ext-$\mathcal{T}$	n/a	39.42	54.72	n/a	6.015	6.265	n/a	479.4	470.7
Interp. MGN (Pfaff et al., 2020)	In-$\mathcal{T}$	2.506	4.834	12.77	1.408	1.289	1.333	203.4	210.4	263.3
	Ext-$\mathcal{T}$	n/a	5.922	36.43	n/a	1.287	1.355	n/a	209.8	263.8
Spatial Oracle (n.c)	In-$\mathcal{T}$		*n/a*			n/a			*n/a*	
	Ext-$\mathcal{T}$	*n/a*	*1.296*	*28.58*	*n/a*	*0.003*	*0.119*	*n/a*	*29.46*	*54.53*
MAgNet (Boussif et al., 2022)	In-$\mathcal{T}$	31.51	135.0	243.9	7.804	6.433	1.884	227.9	220.3	**225.8**
	Ext-$\mathcal{T}$	n/a	142.8	255.5	n/a	6.291	1.947	n/a	229.8	**230.6**
Ours	In-$\mathcal{T}$	**0.2019**	**0.1964**	**0.4062**	**0.4115**	**0.4278**	**0.4549**	**108.0**	**106.1**	278.6
	Ext-$\mathcal{T}$	n/a	**0.2138**	**11.36**	n/a	**0.4326**	**0.4802**	n/a	**119.9**	306.9

Table 7.2: **Time Continuity** – we evaluate the time interpolation power of our method vs. the baselines and standard interpolation techniques. Models are trained and evaluated with only 25% of the full observation grid, and with different temporal resolutions (full, half, and quarter of the original). The Spatial Oracle (*non-comparable!*) uses the exact solution at every point in space, and performs temporal interpolation. Evaluation is conducted over 20 frames in the future (10 for *Eagle*) and we report MSE compared to the ground truth solution ($\times 10^{-3}$).

ena, it would be interesting to add physics priors in the system, a nontrivial extension that we leave for future work. A website has been created where results can be visualized and interact with:

https://continuous-pde.github.io/

7.5 Post-scriptum: taking a step back

This project is the last one conducted during this Ph.D, hence its connection with most of our previous work. The proposed approach shows outstanding performances compared to the state-of-the-art on very challenging datasets while achieving all our requirements stated in the introduction of the chapter. However, there is still room for improvement, in particular concerning the underlying hypotheses required for training the model.

Real world datasets – One might argue that our approach has been tested solely on artificial tasks. Theoretically, using a real-world dataset is possible, but faces several challenges related to the performance evaluation. Indeed, while the model can be trained from sparse data, the evaluation step requires dense observations of the physical simulation to measure the generalization capabilities of the model. Publicly available datasets of real-world physical phenomena that are both large-scale enough to allow for deep learning, and dense enough to allow for good evaluation conditions are rare.

Sampling time during training – we assumed that the sampled time steps in $\mathcal{T}$ are evenly spaced in $[\![0, t]\!]$. This requirement is used for the auto-regressive module which leverages the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

observed states $\boldsymbol{s}_d[n]$ at several instants $n\Delta$. It would be interesting to extend the model to handle irregular time sampling. A possible direction would be to leverage a continuous time system rather than a discrete setup and solve for different times in $\mathcal{T}$ using a Neural-ODE setup.

On one hand, we could still compute $\boldsymbol{z}_d[n]$ at static time steps, but also simulate $\boldsymbol{z}(t)$ for all $t \in \mathcal{T}$, for supervision purposes. However, using a neural-ODE with a simple static method (e.g. runge-kutta) might not be sufficient, as the duration between two times in $\mathcal{T}$ can be large, hence the simulation less precise. Better performance might be obtained with adaptive methods, but this will strongly impact the training time. Finally, the transition to non-uniform $\mathcal{T}$ is not straightforward.

About the GRU – The role of the GRU in our dynamical interpolator is to behave as an observer (see chapter 2 for a reminder). The network starts with a random initial condition $\boldsymbol{u}[0]$ and must forecast the state using the conditioned vectors $\boldsymbol{z}_{n\Delta}(\boldsymbol{x}, t)$. This behavior can be related to a contraction property. This could motivate the use of observers which are provably contractive, such as Deep-KKL (chapter 5).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Part IV

Scaling up: large-scale learning of complex physics phenomena

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

General remarks

Throughout the previous chapters, we noticed that our data greediness is growing bigger and bigger. In Part II, we focused on principled learning-based models and theoretical considerations. These projects are better evaluated with simple and well understood systems, allowing easier analysis of the behavior of a model. In Part III, we addressed more and more challenging problems, mainly under the form of PDEs. In particular, chapter 7 was about fluid dynamics, an infamously difficult simulation task that requires a large amount of training data to perform. In general, we spotted a lack of physics-oriented large-scale datasets in the community, slowing down research towards fast neural simulators. Hence, part of our work focused in providing large-scale and public datasets for physics simulations, as well as strong baseline models to compete with.

Chapter 8: Large-scale learning of turbulent fluid dynamics with mesh transformers

TLDR; Industrial-grade fluid simulations are traditionally riddled by computationally intensive numerical models. We propose a leap in difficulty and performance by introducing Eagle, a new model, method and benchmark for fluid flow forecasting. Our approach relies on a new mesh transformer, leveraging node clustering, graph pooling and global attention to model long-range flow interactions, which were traditionally carried on by multiple GNN iterations. To strengthen our work, we also present a new accurately simulated fluid dynamics dataset that is significantly more challenging than existing tasks in the literature.

Chapter 9: Unsupervised learning of counterfactual physics in pixel space

TLDR; Causal relationships are at the core of physics, yet learning them, especially in high-dimensional data is a hard task, as they are often defined on low-dimensional manifolds and must be extracted from complex signals. We present a method and dataset for learning counterfactual physical processes in pixel space. Going beyond the identification of structural relationships, we deal with the challenging problem of forecasting raw video over long horizons. Our model learns and acts on a suitable hybrid latent representation based on a combination of dense features, sets of 2D keypoints and an additional latent vector per keypoint.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 8

Large-scale learning of turbulent fluid dynamics with mesh transformers

Work presented at ICLR (poster) 2023,
Co-authors: Aurélien Béneteau (SupAero, Toulouse)
Madiha Nadri (Université Lyon 1),
Julie Digne (CNRS, Lyon),
Nicolas Thome (Université Sorbonne)
Christian Wolf (NaverLabs Europe)

8.1 Context

Despite consistently being at the center of attention of mathematics and computational physics, solving the Navier-Stokes equations governing fluid mechanics remains an open problem. In the absence of an analytical solution, fluid simulations are obtained by spatially and temporally discretizing differential equations, for instance with the FVM or FEM. These algorithms are computationally intensive, and take up to several weeks for complex problems while requiring expert configurations.

Neural network-based physics simulators may represent a convenient substitute in many ways. Beyond the expected speed gain, their differentiability would allow for direct optimization of fluid mechanics problems (airplane profiles, turbulence resistance, etc.), opening the way to replace traditional trial-and-error approaches. They would also be an alternative for solving complex PDEs where numerical resolution with conventional tools is intractable. Yet, the development of such models is slowed down by the difficulty of collecting data in sufficient amounts to reach generalization.

Indeed, fluid datasets for deep learning are challenging to produce in many ways. Real-world measurement is complicated, requiring complex velocimetry devices (Wang et al., 2020; Discetti and Coletti, 2018; Erichson et al., 2020). Remarkably, Eckert et al. (2019); Bézenac et al. (2019) leverage alignment with numerical simulation to extrapolate precise ground truth flows on real-world phenomena (smoke clouds and sea surface temperature). Fortunately, accurate simulation data can be acquired through several solvers, ranging from computer graphics-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

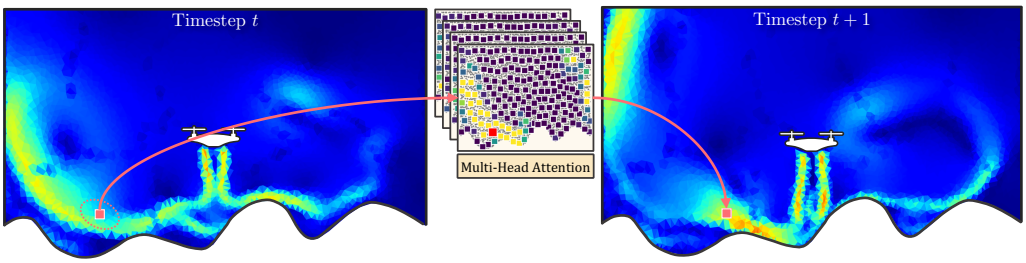


Figure 8.1: **Teaser** – We introduce *Eagle*, a large-scale dataset for learning complex fluid mechanics, accurately simulating the airflow created by a 2D drone in motion and interacting with scenes of varying 2D geometries. We address the problem through an autoregressive model and self-attention over tokens in a coarser resolution, allowing to integrate long-range dependencies in a single hop, shown in the given example by the attention distributions for ■, which follows the airflow.

oriented simulators (Takahashi et al., 2021; Pfaff and Thuerey, 2016) to accurate computational fluid dynamics solver (OpenFOAM©, Ansys© Fluent, ...). A large body of work (Chen et al., 2021a; Pfaff et al., 2020; Han et al., 2021; Stachenfeld et al., 2021) introduces synthetic datasets limited to simple tasks, such as 2D flow past a cylinder.

In this chapter, we introduce *Eagle*, a large-scale dataset for learning unsteady fluid mechanics. We accurately simulate the airflow produced by a two-dimensional Unmanned Aerial Vehicle (UAV) moving in 2D environments with different boundary geometries. This choice has several benefits. It models the complex ground effect turbulence generated by the airflow of an UAV following a control law, and, up to our knowledge, is thus significantly more challenging than existing datasets. It leads to highly turbulent and non-periodic eddies and high flow variety, as the different scene geometries generate completely different outcomes. At the same time, the restriction to a 2D scene (similar to existing datasets) makes the problem manageable and allows for large-scale amounts of simulations (∼1.1m meshes). For a comparison with existing datasets, see table 8.1.

As a second contribution, we propose a new multi-scale attention-based model, which circumvents the quadratic complexity of multi-head attention by projecting the mesh onto a learned coarser representation yielding fewer but more expressive nodes. Conversely to standard approaches based on Graph Neural Networks (GNNs), we show that our model dynamically adapts to the airflow in the scene by focusing attention not only locally, but also over larger distances. More importantly, attention for specific heads seems to align with the predicted airflow, providing evidence of the capacity of the model to integrate long-range dependencies in a single hop (see figure 8.1). We evaluate the method on several datasets and achieve state-of-the-art performance on two public fluid mechanics datasets (Cylinder-Flow, (Pfaff et al., 2020) and Scalar-Flow (Eckert et al., 2019)), and on *Eagle*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Dataset		Size	Public	Dyn. Scene	Dyn. Mesh	# nodes (avg)	# of meas.
Pfaff et al. (2020)	CylinderFlow	15Gb	✓	✗	✗	1,885	0.72M
	AirFoil	56Gb				5,233	0.72M
Stachenfeld et al. (2021)	KS Equation	N.A	✗	✗	✗(Grid)	64	1,200
	Incomp. Dec.					2,304	210
	Comp. Dec.					32,768	35
	Rad. Cooling					32,768	30
Han et al. (2021)	Vascular Flow	N.A.	✗	✗	✓	7,561	5,250
Eckert et al. (2019)	Scalar Flow	351Gb	✓	✗	✗(Grid)	1.7M	0.015M
Bézenac et al. (2019)	SST	N.A	✓	✗	✗(Grid)	4,096	0.1M
Eagle (Ours)		**270Gb**	✓	✓	✓	**3,388**	**1.18M**

Table 8.1: **Comparison with available datasets** – To the best of our knowledge, *Eagle* is the first dataset of such scale, complexity, and variety. Smaller-scale datasets such as Li et al. (2008); Wu et al. (2017b) have been excluded, as they favor simulation accuracy over size. The datasets in Stachenfeld et al. (2021) are not public, but can be reproduced from the information in the paper.

8.2 The Eagle dataset and benchmark

Eagle is comprised of fine-grained fluid simulations defined on irregular triangle meshes, which we argue is more suited to a broader range of applications than regular grids and thus more representative of industrial standards. Compared to grid-based datasets (Bézenac et al., 2019; Stachenfeld et al., 2021), irregular meshes provide better control over the spatial resolution, allowing for finer discretization near sensitive areas. This property is established for most fluid mechanics solvers (Versteeg and Malalasekera, 2007) and seems to transfer well to simulators based on machine learning (Pfaff et al., 2020). However, using triangular meshes with neural networks is not as straightforward as regular grids. Geometric deep learning (Bronstein et al., 2021) and graph networks (Battaglia et al., 2018) have established known baselines but this remains an active domain of research. Existing datasets focus on tasks such as the flow past an object (Chen et al., 2021a; Pfaff et al., 2020) or turbulent flow on an airfoil (Thuerey et al., 2020; Sekar et al., 2019). These are well-studied problems, for some of which analytical solutions exist, and they rely on a large body of work from the physics community. However, the generated flows, while being turbulent, are merely steady or periodic despite variations in the geometry. With *Eagle*, we propose a complex task, with convoluted, unsteady, and turbulent airflow with minimal resemblance across each simulation.

Purpose – we built *Eagle* to meet a growing need for a fluid mechanics dataset in accordance with the methods used in engineering, i.e. reasoning on irregular meshes. To significantly increase the complexity of the simulations compared to existing datasets, we propose a proxy task consisting of studying the airflow produced by a dynamically moving UAV in many scenes with variable geometry. This is motivated by the highly non-steady turbulent outcomes that this task generates, yielding challenging airflow to be forecasted. Particular attention has

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

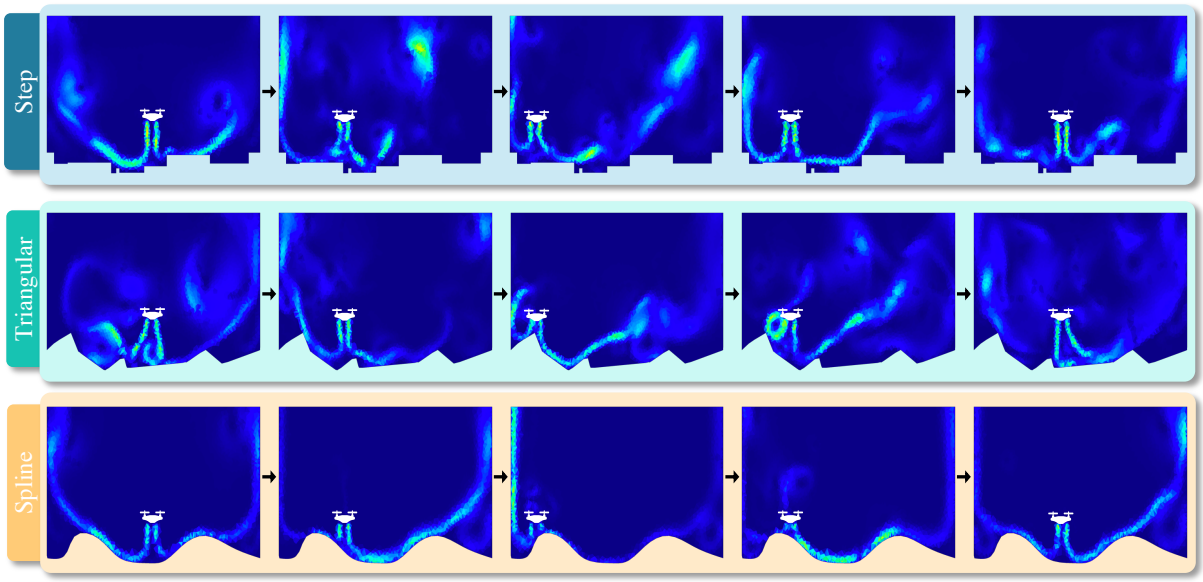


Figure 8.2: **Snapshots from the dataset** – Velocity field norm over time for three episodes, one for each geometry type. Turbulence is significantly different from one simulation to another and strongly depends on the ground surface.

also been paid to the practical usability of *Eagle* with respect to the state-of-the-art in fluid dynamics neural simulators by controlling the number of mesh points and limiting confounders variables to a moderate amount (i.e. scene geometry and drone trajectory).

Simulation and task definition – we simulate the complex airflow generated by an UAV maneuvering in 2D scenes with varying floor profiles. While the scene geometry varies, the UAV trajectory is constant: it starts in the center of the scene and navigates, hovering near the floor surface. During the flight, the two propellers generate high-paced airflows interacting with each other and with the structure of the scene, causing convoluted turbulence. To produce a wide variety of different outcomes, we procedurally generate a large number of floor profiles by interpolating a set of randomly sampled points within a certain range. The choice of interpolation order induces drastically different floor profiles, and therefore distinct outcomes from one simulation to another. *Eagle* contains three main types of geometry depending on the type of interpolation[1] (see Figure 8.2):

- **Step**: surface points are connected using step functions (zero-order interpolation), which produce very stiff angles with drastic changes of the airflow when the UAV hovers over a step.
- **Triangular**: surface points are connected using linear functions (first-order interpolation), causing the appearance of many small vortices at different locations in the scene.
- **Spline**: surface points are connected using spline functions with smooth boundaries, causing long and fast trails of air, occasionally generating complex vortices.

[1]Videos on the project website https://eagle-dataset.github.io

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Eagle contains about 600 different geometries (200 geometries of each type) corresponding to roughly 1,200 flight simulations (one geometry gives two flight simulations depending on whether the drone is going to the right or the left of the scene), performed at 30 fps over 33 seconds, resulting in 990 timesteps per simulation. Physically plausible UAV trajectories are obtained through MPC control of a (flow agnostic) dynamical system we design for a 2D drone. More details and statistics are available in appendix E.1.

We simulated the temporal evolution of the velocity field as well as the pressure field (both static and dynamic) defined over the entire domain. Due to source motion, the triangle mesh on which these fields are defined needs to be dynamically adapted to the evolving scene geometry. More formally, the mesh is a valued dynamical graph $\mathcal{M}^t = (\mathcal{X}^t, \mathcal{E}^t, \mathcal{V}^t, \mathcal{P}^t; \mathcal{N}^t)$ where $\mathcal{X} \subset \mathbb{R}^2$ is the set of nodes positions, $\mathcal{E} \subset (\mathbb{N} \times \mathbb{N})$ the edges pairs, $\mathcal{V} \subset \mathbb{R}^2$ is a field of velocity vectors and $\mathcal{P} \subset \mathbb{R}$ is a field of scalar pressure values. Both physical quantities are expressed at node level. The set $\mathcal{N}$ contains node type attributes, indicating if a node belongs to a wall, an input, or an output boundary. Note that the dynamical mesh is completely flow-agnostic, thus no information about the flow can be extrapolated directly from the future node positions. Time dependency will be omitted when possible for the sake of readability.

Numerical simulations – were carried out using the software Ansys© Fluent, which solves the RANS equations with the Reynolds stress model. It uses five equations to model turbulence, a more accurate approach than standard k-ϵ or k-ω models (two equations). This resulted in 3.9TB of raw data with ∼162,760 control points per mesh. We down-sampled this to 3,388 points on average and compressed it to 270GB. Details and illustrations are given in appendix E.1.

Task – for what follows, we define $\boldsymbol{x}_i \in \mathcal{X}^t$ as the 2D position of node i, $\boldsymbol{v}_i \in \mathcal{V}^t$ its velocity, $p_i \in \mathcal{P}^t$ pressure and $n_i \in \mathcal{N}^t$ its node type. We are interested in the following task: given the complete simulation state at time t, namely $\mathcal{M}^t$, as well as future mesh geometry $\mathcal{X}^{t+h}, \mathcal{E}^{t+h}$, forecast the future velocity and pressure fields $\mathcal{V}^{t+h}, \mathcal{P}^{t+h}$, i.e. for all positions i we predict $\boldsymbol{v}_i^{t+h}, p_i^{t+h}$ over a horizon h. Importantly, we consider the dynamical re-meshing step $\mathcal{X}^t \rightarrow \mathcal{X}^{t+h}$ to be known during inference and thus is not required to be forecasted.

8.3 Learning unsteady airflow

Accurate flow estimations require data on a certain minimum spatial and temporal scale. Deviations from optimal resolutions, i.e. data sampled with lower spatial resolutions or lower frame rates, are typically very hard to compensate through models of higher complexity, in particular when the estimation is carried out through numerical simulations with an analytical model. The premise of our work is that machine learning can compensate for loss in resolution by picking up longer rate regularities in the data, trading data resolution for complexity in the modeled interactions. Predicting the outcome for a given mesh position may therefore require

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

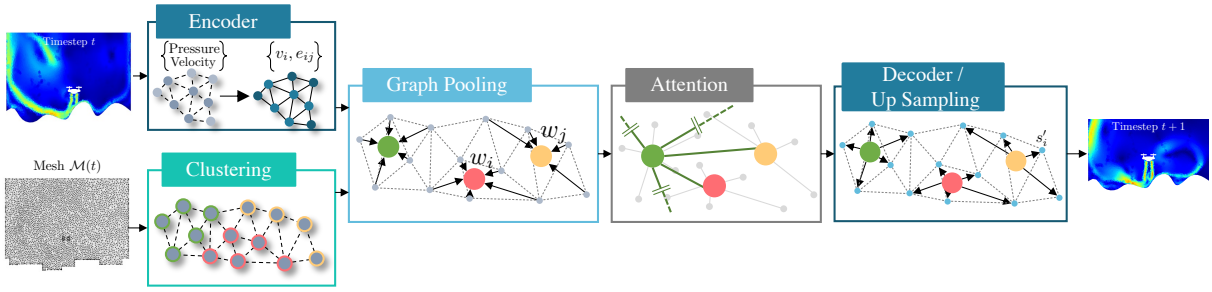


Figure 8.3: **Model overview** – The mesh transformer encodes the input mesh node values (positions, pressure, and velocity), reduces the spatial resolution through clustering + graph pooling and performs multi-head self-attention on the coarser level of cluster centers. A decoder upsamples the token embeddings to the original resolution and predicts pressure and velocity at time step $t + 1$.

information from a larger neighborhood, whose size can depend on factors like resolution, compressibility, Reynolds number, etc.

Regularities and interactions on meshes and graphs have classically been modeled with probabilistic graphical models (MRFs (Geman and Geman, 1984), CRFs (Lafferty et al., 2001), RBMs (Smolensky, 1986) etc.), and in the deep learning era through geometric DL (Bronstein et al., 2021) and graph networks (Battaglia et al., 2018), or deep energy-based models. These models can capture long-range dependencies between distant nodes but need to exploit them through multiple iterations. In this work, we argue for the benefits of transformers and self-attention (Vaswani et al., 2017), which in principle are capable of integrating long-range interactions in a single step.

However, the quadratic complexity of transformers in terms of number of tokens makes its direct application to large meshes expensive. While low-complexity variants do exist (Katharopoulos et al., 2020), we propose a different *Ansatz*, shown in figure 8.3. We propose to combine graph clustering and learned graph pooling to perform full attention on a coarser scale with higher-dimensional node embedding. This allows the dot-product similarity of the transformer model – which is at the heart of the crucial attention operations – to operate on a semantic representation instead of on raw input signals, similar to the settings in other applications. In natural language processing, attention typically operates on word embeddings (Vaswani et al., 2017), and in vision either on patch embeddings (Dosovitskiy et al., 2021) or on convolutional feature map cells (Wang et al., 2018). In the sequel, we present the main modules of our model (further details are given in appendix E.2).

Offline Clustering – we down-scale mesh resolution through geometric clustering, which is independent of the forecasting operations and therefore pre-computed offline. A modified k-means clustering is applied to the vertices $\mathcal{X}^t$ of each time step and creates clusters with a constant number of nodes (details are given in appendix E.2.1). The advantages are twofold: (a) the irregularity and adaptive resolution of the original mesh are preserved, as high-density regions will require more clusters, and (b) constant cluster sizes facilitate parallelization and

allow to speed up computations. In what follows, let $\mathcal{C}_k$ be the kth cluster computed on mesh $\mathcal{M}^t$.

Encoder – the initial mesh $\mathcal{M}^t$ is converted into a graph $\mathcal{G}$ using the encoder in Pfaff et al. (2020). More precisely, node and edge features are computed using MLPs ϕ_{node} and ϕ_{edge}, giving

$$\begin{aligned} \eta_i^1 &= \phi_{\text{node}}(\boldsymbol{v}_i, p_i, n_i) && \forall (\boldsymbol{v}_i, p_i, n_i) \in \mathcal{V} \times \mathcal{P} \times \mathcal{N}, \\ e_{ij}^1 &= \phi_{\text{edge}}(\boldsymbol{x}_i - \boldsymbol{x}_j, |\boldsymbol{x}_i - \boldsymbol{x}_j|) && \forall (i,j) \in \mathcal{E} \end{aligned} \tag{8-1}$$

where $|\cdot|$ is the euclidean norm. The encoder also computes an appropriate positional encoding based upon spectral projection $F(\boldsymbol{x})$. We also leverage the local position of each node in its cluster. Let $\bar{\boldsymbol{x}}_k$ be the barycenter of cluster $\mathcal{C}_k$, then the local encoding of node i belonging to cluster k is the concatenation $\boldsymbol{f}_i = \left[F(\boldsymbol{x}_i)^\top \quad F(\bar{\boldsymbol{x}}_k - \boldsymbol{x}_i)^\top \right]^\top$. Finally, a series of L GNNs extracts local features through message passing

$$\begin{aligned} e_{ij}^{\ell+1} &= e_{ij}^\ell + \overbrace{\psi_{\text{edge}}^\ell \left(\left[\eta_i^\ell \ \boldsymbol{f}_i \right], \left[\eta_j^\ell \ \boldsymbol{f}_j \right], e_{ij}^\ell \right)}^{\varepsilon_{ij}}, \\ \eta_i^{\ell+1} &= \eta_i^\ell + \psi_{\text{node}}^\ell \left(\left[\eta_i^\ell \ \boldsymbol{f}_i \right], \sum_j \varepsilon_{ij} \right). \end{aligned} \tag{8-2}$$

The superscript ℓ indicates the layer, and ψ_{node}^ℓ and ψ_{edge}^ℓ are MLPs which encode nodes and edges, respectively. The exact architecture hyper-parameters are given in appendix E.2. For the sake of readability, in what follows, we will note $\eta_i = \eta_i^L$ and $e_{ij} = e_{ij}^L$.

Graph Pooling – summarizes the state of the nodes of the same cluster $\mathcal{C}_k$ in a single high-dimensional embedding $\boldsymbol{w}_k$ on which the main neural processor will reason. This is performed with a Gated Recurrent Unit (GRU) (Cho et al., 2014) where the individual nodes are integrated sequentially in random order. This allows to learn a more complex integration of features than a sum. Given an initial GRU state $\boldsymbol{u}^0 = 0$, node embeddings are integrated iteratively, indicated by superscript n,

$$\begin{aligned} \boldsymbol{u}_k^{n+1} &= r_{\text{gru}} \left(\left[\eta_i \ \boldsymbol{f}_i \right], \boldsymbol{u}_k^n \right), \quad i \in \mathcal{C}_k, \\ \boldsymbol{w}_k &= \phi_{\text{cluster}}(\boldsymbol{u}_k^N), \end{aligned} \tag{8-3}$$

where $N = |\mathcal{C}_k|$ and ϕ_{cluster} is an MLP. r_{gru} denotes the update equations of a GRU, where we omitted gating functions from the notation. The resulting set of cluster embeddings $\mathcal{W} = \left\{ \boldsymbol{w}_k \,|\, k \in [\![1, K]\!] \right\}$ significantly reduces the spatial complexity of the mesh.

Attention Module – consists of a Transformer with M layers of multi-head attention (MHA) (Vaswani et al., 2017) working on the embeddings $\mathcal{W}$ of the coarse graph. Setting $\boldsymbol{w}_k^1 = \boldsymbol{w}_k$, we get for layer m:

$$\boldsymbol{w}_k^{m+1} = f_{\text{mha}} \Big(\text{Q} = \left[\boldsymbol{w}_k^m \ F(\bar{\boldsymbol{x}}_k) \right], \text{K} = \text{V} = \mathcal{W} \Big), \tag{8-4}$$

where Q, K, and V are, respectively, the *query*, *key*, and *value* mappings of a Transformer. We refer to Vaswani et al. (2017) for the details of multi-head attention, denoted as f_{mha}.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Decoder – the output of the attention module is calculated on the coarse scale, with one embedding per cluster. The decoder upsamples the representation and outputs the future pressure and velocity field on the original mesh. This upsampling is done by taking the original node embedding η_i and concatenating with the cluster embedding $\boldsymbol{w}_k^M$, followed by the application of a GNN, whose role is to take the information produced on a coarser level and correctly distribute it over the nodes i. To this end, the GNN has access to the positional encoding of the node, which is also concatenated

$$\begin{cases} \hat{\boldsymbol{v}}^{t+1} &= \boldsymbol{v}^t + \delta_v \\ \hat{p}^{t+1} &= p^t + \delta_p \end{cases}, \ (\delta_v, \delta_p) = g_{\text{gnn}}\left([\eta_i \ \boldsymbol{w}_k^M \ \boldsymbol{f}_i]\right), \tag{8-5}$$

where $i \in \mathcal{C}_k$ and g_{gnn} is the graph network variant described in equation (8-2), parameters are not shared. Our model is trained end-to-end, minimizing the forecasting error over horizon H (potentially lower than the target horizon h used during the test phase) where α balances the importance of the pressure field over the velocity field

$$\mathcal{L} = \sum_{i=1}^{H} \text{MSE}\left(\boldsymbol{v}^{t+i}, \hat{\boldsymbol{v}}^{t+i}\right) + \alpha \sum_{i=1}^{H} \text{MSE}\left(p^{t+i}, \hat{p}^{t+i}\right). \tag{8-6}$$

8.4 Experiments

We compare our method against three competing methods for physical reasoning:

- **MeshGraphNet** (Pfaff et al., 2020) (MGN) is a GNN-based model that relies on multiple chained message passing layers.
- **Graph Attention Transformer** (Veličković et al., 2017) (GAT) is based upon MGN where the GNNs interactions are replaced by graph attention Transformers. Compared to our mesh transformer, here attention is computed over the one-ring of each node only.
- **DilResNet** (Stachenfeld et al., 2021) (DRN) differs from the other models as it does not reason over nonuniform meshes, but instead uses dilated convolution layers to perform predictions on regular grids.

To evaluate the latter on *Eagle*, we interpolate grid-based simulation over the original mesh (see appendix E.1.2). During validation and testing, we project airflow back from the grid to the original mesh in order to compute comparable metrics. All baselines have been adapted to the dataset using hyperparameter sweeps, which mostly lead to increases in capacity, explained by the complexity of *Eagle*. We also compare to two other datasets:

- **Cylinder-Flow** (Pfaff et al., 2020) simulates the airflow behind a cylinder with different radii and positions. This setup produces turbulent yet periodic airflows corresponding to Karman vortex.
- **Scalar-Flow** (Eckert et al., 2019) contains real world measurements of smoke cloud. This dataset is built using velocimetry measurements combined with numerical simulation aligned with the observations. Following Lienen and Günnemann (2022); Kohl

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

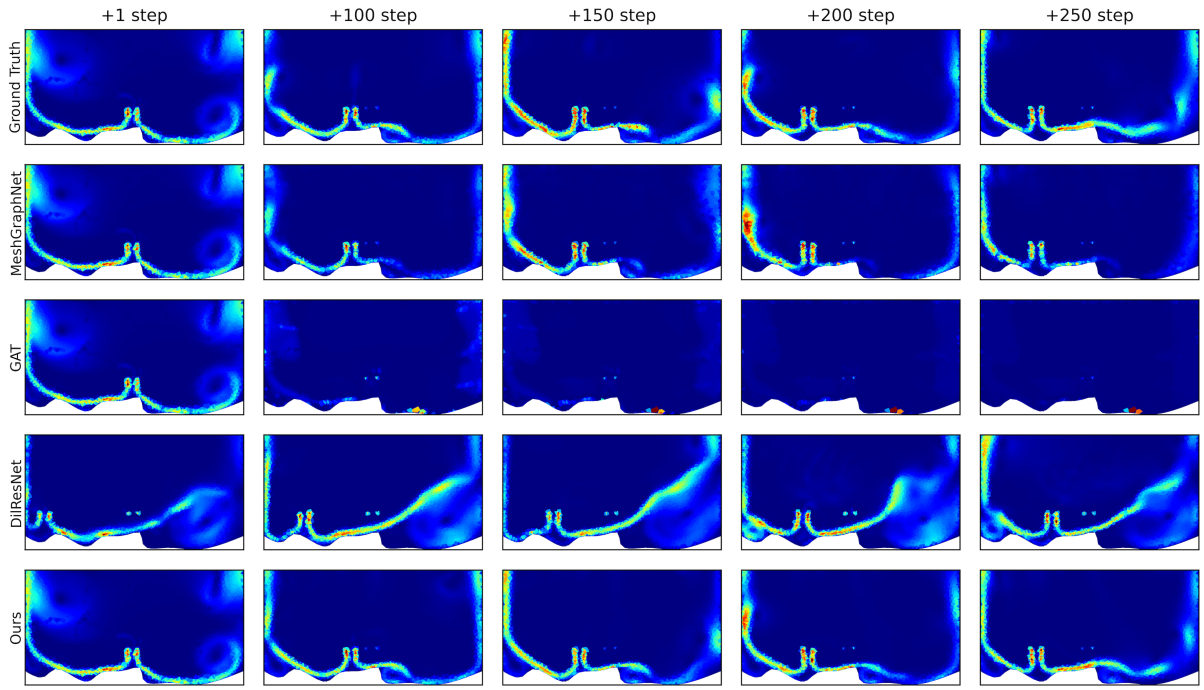


Figure 8.4: **Qualitative comparisons** – with the state of the art on *Eagle*. We color code the norm of the velocity field.

Dataset	Cylinder Flow			Scalar Flow			*Eagle*		
Horizon	+1	+50	+250	+1	+50	+100	+1	+50	+250
MeshGraphNet	0.0058	0.0405	0.0792	0.0374	0.3193	0.5944	0.0916	0.5698	0.9896
GAT	0.0112	0.1774	1.3336	0.0434	0.3991	0.6414	7.5587	19.260	26.867
DilResNet	0.0429	0.0626	0.1295	0.0372	0.2212	0.3975	0.2987	0.5650	0.8944
Ours	**0.0030**	**0.0221**	**0.0735**	**0.0128**	**0.0869**	**0.1467**	**0.0811**	**0.3495**	**0.6357**

Table 8.2: **Quantitative results** – Norm. RMSE (velocity and pressure) for our model (cluster size = 10) and the baselines.

et al. (2020), we reduce the data to 2D grid-based simulation by averaging along the x-direction.

We evaluate all models reporting the sum of the root mean squared error (N-RMSE) on both pressure and velocity fields, which have been normalized with regards to the training set (centered and reduced), and we provide finer-grained metrics in appendix E.3.1.

Existing datasets – show little success in discriminating the performances of fluid mechanics models (see table 8.2). On Cylinder-Flow, both ours and MeshGraphNet reach near-perfect forecasting accuracy. Qualitatively, flow fields are hardly distinguishable from the ground truth at least for the considered horizon (see appendix E.3.2). As stated in the previous sections, this dataset is a great task to validate fluid simulators but may be considered as saturated. Scalar-Flow is a much more challenging benchmark, as these real-world measurements are limited in resolution and quantity. Our model obtains good quantitative results, especially

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Ablation	**N-RMSE** (+250)	
Clustering	1 node per cluster	20 nodes per cluster
GNN	–	1.3484
One-ring	1.0258	0.7976
Average	0.7876	0.7797
Ours	–	**0.6572**

Table 8.3: **Ablations** – GNN replaces global attention by a set of L GNNs on the coarser mesh. *One-ring* constrains attention to the one-ring. *Average* forces uniform attention.

on a longer horizon, showing robustness to error accumulation during auto-regressive forecasting. Yet, no model achieved visually satisfactory results, the predictions remain blurry and leave room for improvements (cf figure in appendix).

Comparisons with the state-of-the-art – are more clearly assessed on *Eagle*. Our model gives excellent results and outperforms competing baselines. It succeeds in forecasting turbulent eddies even after a long prediction horizon. Our model outperforms MeshGraphNet, which provides evidence for the interest in modeling long-range interactions with self-attention. GAT seems to struggle with our challenging dataset. The required increase in capacity was difficult to do for this resource-hungry model, we failed even on the 40GB A100 GPUs of a high-end Nvidia DGX.

DilResNet shows competitive performances on *Eagle*, consistent with the claims of the original paper. However, it fails to predict details of the vortices (cf. Figure 8.4). This model leverages grid-based data and was trained on a voxelled simulation finally projected back on the triangular mesh during testing. This requires precaution in assessment. We try to limit projection error by setting images to contain ten times more pixels than nodes in the actual mesh. Yet, even at that scale, we measure that the reconstruction error represents roughly a third of the final N-RMSE. This points out that grid-based simulations are not suited for complex fluid problems such as *Eagle*, which require finer spatial resolution near sensitive areas. We expose failure cases in appendix E.3.3.

Self-attention – is a key feature in our model, as shown in Figures 8.5b and c, which plot the gradient intensity of a selected predicted point □ situated on the trail with regards to all input points, for fixed trained model weights. MeshGraphNet is inherently limited to a neighborhood determined by the number of chained GNNs, the receptive field, which is represented as concentric black circles overlaid over the gradients. In contrast, our model is not spatially limited and can exchange information across the entire scene, even possibly in a single step. The gradients show that this liberty is exploited.

In the same figure we also show the attention maps, per head and layer, for the selected point □ near the main trail in Figure 8.5d. Interestingly, our model discovers to attend not only to the neighborhood (as a GNN would) but also to much farther areas. More importantly, we

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

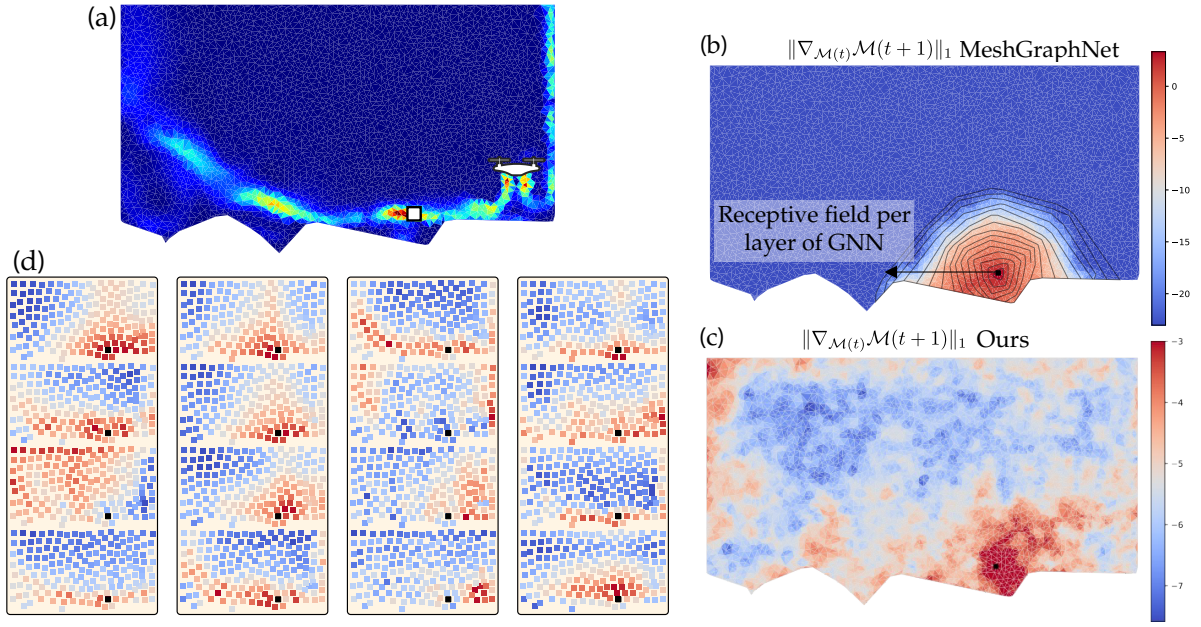


Figure 8.5: **Locality of reasoning**. (a) Velocity of an example flow and a selected point □; (b): The receptive field for this point for the MeshGraphNet model (Pfaff et al., 2020) is restricted to a local neighborhood, also illustrated through the overlaid gradients $||\nabla_{\mathcal{M}^t}\mathcal{M}^{t+1}||_1$. (c): the receptive field of our method covers the whole field and the gradients indicate that this liberty is exploited; (d) the attention distributions for point □, certain maps correlate with airflow. Attention maps can be explored interactively using the online tool at https://eagle-dataset.github.io.

observe that certain heads explicitly (and dynamically) focus on the airflow, which provides evidence that attention is guided by the regularities in the input data. We released an online tool allowing interactive visualization and exploration of attention and predictions, available at https://eagle-dataset.github.io.

Ablation studies – indicate how global attention impacts performance: (a) closer to MeshGraphNet, we replace the attention layers by GNNs operating on the coarser mesh, allowing message passing between nearest cluster only; (b) we limit the receptive field of MHA to the **one-ring** of the cluster; (c) we enforce uniform attention by replacing it with an **average** operation. As shown in table 8.3, attention is a key design choice. Disabling attention to distant points hurts RMSE, indicating that the model leverages efficient long-range dependencies. Agnostic attention to the entire scene is not pertinent either: to be effective, attention needs to dynamically adapt to the predicted airflow. We also conduct a study on generalization to down-sampled meshes in appendix E.3.4.

The role of clustering – is to summarize a set of nodes into a unique feature vector. Arguably, with bigger clusters, more node-wise information must be aggregated in a finite-dimensional vector. We indeed observe a slight increase in N-RMSE when the cluster size increases (Figure 8.6a). Nonetheless, our model appears to be robust to even aggressive graph clustering as the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

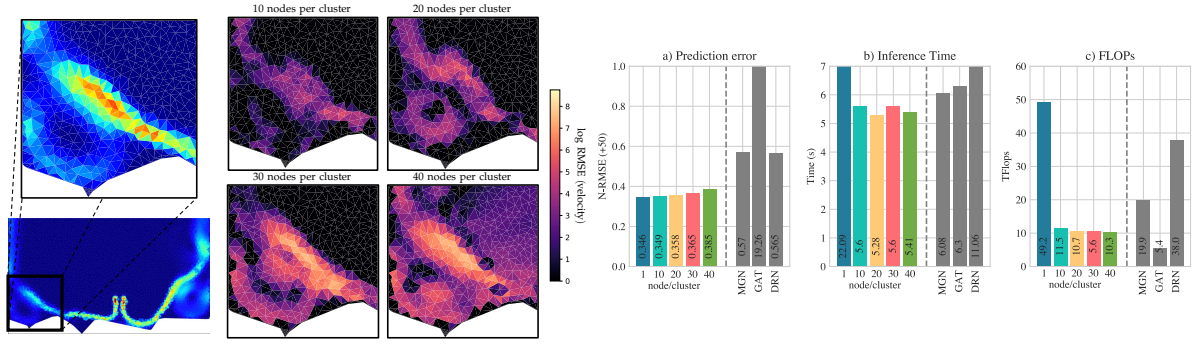


Figure 8.6: **Impact of cluster size** – *Left*: We color code RMSE in logarithmic scale on the velocity field near a relatively turbulent area at a horizon of h=400 steps of forecasting. *Right*: Error (+50 timesteps), inference time, and FLOPs for different cluster sizes and the baselines.

Model		Ours				MGN			
Ablated Geometry		Stp	Spl	Tri	∅	Stp	Spl	Tri	∅
N-RMSE (+250)	Stp	0.927	0.865	1.132	**0.828**	2.062	1.236	1.347	1.116
	Spl	0.595	0.584	0.857	**0.488**	1.257	0.941	1.037	0.807
	Tri	0.730	0.732	1.049	**0.647**	1.685	1.100	1.131	1.037

Table 8.4: **Generalization to unseen geometries** – we evaluate our model and MeshGraphNet in different setups, evaluating all geometry types but removing one from training. Our model shows satisfactory generalization, highlighting the complementarity of each simulation type.

drop remains limited and still outperforms the baselines. A qualitative illustration is shown in figure 8.6 (left), where we simulate the flow up to 400 time-steps forward and observe the error on a relatively turbulent region. Clustering also acts on the complexity of our model by reducing the number of tokens on which attention is computed. We measure a significant decrease in inference time and number of operations (FLOPs) even when we limit clusters to a small size (Figure 8.6b and c).

Generalization experiments – highlight the complementarity of the geometry types in *Eagle* since the removal of one geometry in the training set impacts the performances on the others. MeshGraphNet suffers the most from this ablation, resulting in a drop ranging from 10% on average (ablation of Spline) to 67% (ablation of Step). In our model, the performance drops are limited for the ablation of Step and Spline. The most challenging geometry is arguably Triangular, as the ground profile tends to generate more turbulent and convoluted flows.

8.5 Conclusion

We presented a new large-scale dataset for deep learning in fluid mechanics. *Eagle* contains accurate simulations of the turbulent airflow generated by a flying drone in different scenes. Simulations are unsteady, highly turbulent, and defined on dynamic meshes, which represents a real challenge for existing models. To the best of our knowledge, we released the first publicly available dataset of this scale, complexity, precision, and variety. We proposed a new

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

model leveraging mesh transformers to efficiently capture long-distance dependencies on a coarser scale. Through graph pooling, we show that our model reduces the complexity of multi-head attention and outperforms the competing state-of-the-art on both existing datasets and *Eagle*. We showed across various ablations and illustrations that global attention is a key design choice and observed that the model naturally attends to airflow.

8.6 Post-scriptum: taking a step back

Before diving into perspective and feedback on this work, I would like to highlight the tremendous implication of Aurélien Béneteau, intern with our team during his Master's degree. Aurélien took care of the generation of the *Eagle* dataset, which represents an enormous amount of work. This project could not have been done without his participation. This dataset is a leap forward in difficulty with regards to existing baselines, and the many challenges in generating such amount of simulations are only partially mentioned in the previous sections. However, there is still room for future work.

Toward 3D simulations – While we think that 3D simulations are indeed the long-term future on this subject, the complexity in factors of variation we need for large-scale machine learning is currently not possible in 3D simulations. Up to date, fluids datasets in 3D are very limited, due to the computation time required for simulation. Classical workarounds rely on relaxing physical accuracy or versatility of the solver, e.g. with SPH simulations. Accurate 3D simulations are mostly conducted on grid-based meshes and for rather simple, theoretic problems (Mohan et al., 2020; Chen et al., 2021b; Stachenfeld et al., 2021). The John Hopkins Turbulent Database (Li et al., 2008) contains nine direct numerical simulation datasets (i.e. direct resolution of Navier-Stokes equations) but with only a single scene per dataset simulated on a very fine grid and low time resolution.

Training neural networks on 3D simulations is also very challenging. *Eagle* already plays with the limit of what neural simulators can handle on decent hardware (40GB of GPU memory is a bare minimum to train our model on *Eagle* in a reasonable amount of time). Extending to 3D would require more computing power than nowadays available to most researchers. Yet, GNN-based models are theoretically not restricted to 2D and can readily manage 3D simulations.

Improvements on the model architecture – Our mesh transformer reasons over two different scales to perform prediction: the finer one with GNN and the coarser one with attention mechanisms. It has been shown that multi-scale networks are beneficial for physics-related tasks (Ronneberger et al., 2015; Cao et al., 2022; Lino et al., 2021). A natural track for improvement could be to consider more scales in our mesh transformer using successive layers of clustering.

Another lead for improvement is to integrate more physics-informed mechanisms in the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

simulator. The Navier-Stokes equations are general enough to be considered as a general-purpose prior. Moreover, most fluid simulations are conducted in an incompressible framework (for flow velocities roughly below 0.5 Mach). This can be enforced in the model by forecasting a scalar value and modeling the velocity field as its rotational, as suggested in Wandel et al. (2021).

Adaptability – The main issue of this area of research is the lack of adaptability. Neural simulators such as ours or MeshGraphNet trained on Cylinder-flow will excel at forecasting the flow past a cylinder but fail spectacularly on other tasks, such as *Eagle*, and vice-versa. Moreover, a model trained on Eagle may generalize to various geometries, but not to various fluid types, and probably neither to very different flight paths. Adaptability in neural simulators is difficult to achieve on complex tasks. To do so, the neural model is usually integrated into a general-purpose solver, which increases the computational cost (Yin et al., 2021a; Kirchmeyer et al., 2022; Heinonen and Lähdesmäki, 2019).

More versatile models require more diversified datasets, which is another obstacle for general-purpose neural simulators. *Eagle* is limited to two confounders (geometry and flight path), and extending the number of variables implies increasing the size of the dataset to carry a sufficient amount of examples of each configuration. This greatly increases the difficulty of generating such a dataset.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 9

Unsupervised learning of counterfactual physics in pixel space

Work presented at ICLR (oral) 2023,
Co-authors: Fabien Baradel (NaverLabs Europe)
Natalia Neverova (Meta AI),
Madiha Nadri (Université Lyon 1),
Greg Mori (Simon Fraser University)
Christian Wolf (NaverLabs Europe)

9.1 Context

Reasoning on complex, multi-modal and high-dimensional data is a natural ability of humans and other intelligent agents (Martin-Ordas et al., 2008), and one of the most important and difficult challenges of AI. While machine learning is well suited for capturing regularities in high-dimensional signals, in particular by using high-capacity deep networks, some applications also require an accurate modeling of causal relationships. This is particularly relevant in physics, where causation is considered a fundamental axiom. In the context of machine learning, correctly capturing or modeling causal relationships can also lead to more robust predictions, in particular better generalization to out-of-distribution samples, indicating that a model has overcome the exploitation of biases and shortcuts in the training data.

In recent literature on physics-related machine learning, causality has often been forced through the addition of prior knowledge about the physical laws that govern the studied phenomena, e.g. Yin et al. (2021b). A similar idea lies behind structured causal models, widely used in the causal inference community, where domain experts model these relationships directly in a graphical notation. This particular line of work allows to perform predictions beyond statistical forecasting, for instance by predicting unobserved counterfactuals, the impact of unobserved interventions (Balke and Pearl, 1994), i.e. answering the question

What alternative outcome would have happened, if the observed event X had been replaced with an event Y (after an intervention)?

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Counterfactuals are interesting, as causality intervenes through the effective modification of an outcome. As an example, taken from (Schölkopf et al., 2021), an agent can identify the direction of a causal relationship between an umbrella and rain from the fact that removing an umbrella will not affect the weather. We focus on counterfactual reasoning on high-dimensional signals, in particular videos of complex physical processes. Learning such causal interactions from data is a challenging task, as spurious correlations are naturally and easily picked up by trained models. Previous work in this direction was restricted to discrete outcomes, as in *CLEVRER* (Yi et al., 2019), or to the prediction of 3D trajectories, as in *CoPhy* (Baradel et al., 2020), which also requires supervision of object positions.

In this chapter, we address the hard problem of predicting the alternative (counterfactuals) outcomes of physical processes in pixel space, i.e. we forecast sequences of 2D projective views of the 3D scene, requiring the prediction over long horizons (150 frames corresponding to $\sim$ 6 seconds). We conjecture that causal relationships can be modeled on a low dimensional manifold of the data, and propose a suitable latent representation for the causal model, in particular for the estimation of the confounders and the dynamic model itself. Similar to V-CDN (Kulkarni et al., 2019; Li et al., 2020a), our latent representation is based on the unsupervised discovery of keypoints, complemented by additional information in our case. Indeed, while keypoint-based representations can easily be encoded from visual input, as stable mappings from images to points arise naturally, we claim that they are not the most suitable representation for dynamic models. We identified and addressed two main problems:

1. The individual points of a given set are discriminated through their 2D positions only, therefore shape, geometry, and relationships between multiple moving objects need to be encoded through the relative positions of points to each other,
2. The optimal representation for a physical model is not necessarily a 2D keypoint space, where the underlying object dynamics has also been subject to the imaging process (projective geometry).

We propose a new counterfactual model, which learns a sparse representation of visual input in the form of 2D keypoints coupled with a (small) set of coefficients per point modeling complementary shape and appearance information. Confounders (object masses and initial velocities) in the studied problem are extracted from this representation, and a learned dynamical model forecasts the entire trajectory of these keypoints from a single (counterfactual) observation. We show, that these design choices are key to the performance of our model and that they significantly improve the capability to perform long-term predictions. Our proposed model outperforms strong baselines for physics-related learning of video prediction.

We introduce a new challenging dataset for this problem, which builds on *CoPhy*, a recent counterfactual physics benchmark (Baradel et al., 2020). We go beyond the prediction of sequences of 3D positions and propose a counterfactual task for predictions in pixel space after

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

interventions on initial conditions (displacing, re-orienting, or removing objects). In contrast to the literature, our benchmark also better controls the identifiability of causal relationships and counterfactual variables and provides more accurate physics simulation.

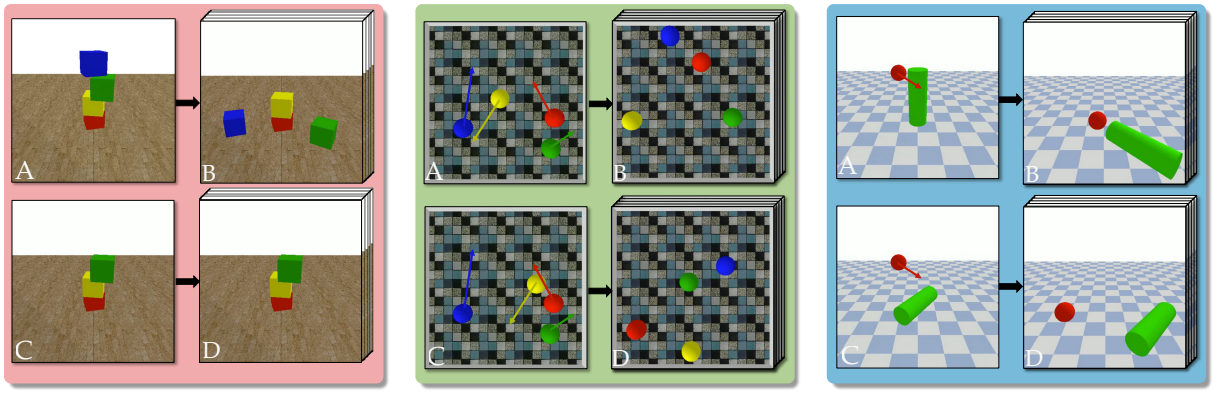
Counterfactual (CF) reasoning and learning of causal relationships in machine learning was made popular by the work of Judas Pearl (Pearl, 2000), which motivates and introduces mathematical tools detailing the principles of *do-calculus*, i.e. study of unobserved interventions on data. A more recent survey links these concepts to the literature in machine learning (Schölkopf et al., 2021). The last years have seen the emergence of several benchmarks for Counterfactual (CF) reasoning in physics. *CLEVRER* (Yi et al., 2019) is a visual question-answering dataset, where an agent is required to answer a CF question after observing a video showing 3D objects moving and colliding. Li et al. (2020a) introduce a CF benchmark with two tasks: a scenario where balls interact with each other according to unknown interaction laws (such as gravity or elasticity), and a scenario where clothes are folded by the wind. The agent needs to identify CF variables and causal relationships between objects and to predict future frames. *CoPhy* (Baradel et al., 2020) clearly dissociates the observed experiment from the CF one, and contains three complex 3D scenarios involving rigid body dynamics. However, the proposed method relies on the supervision of 3D object positions, while our work does not require any metadata.

9.2 The Filtered-CoPhy benchmark

We build on *CoPhy* (Baradel et al., 2020), retaining its strengths, but explicitly focusing on a counterfactual scenario in pixel space and eliminating the ill-posedness of tasks we identified in the existing work. Each data sample is called an *experiment*, represented as a pair of trajectories: an *observed* one with initial condition $X_0 = \mathbf{A}$ and outcome $\{X_t \,|\, \forall t \in [\![0, T]\!]\} = \mathbf{B}$ (a sequence), and a *counterfactual* one $\bar{X}_0 = \mathbf{C}$ and $\{\bar{X}_t \,|\, \forall t \in [\![0, T]\!]\} = \mathbf{D}$ (a sequence). Throughout this paper we will use the letters **A**, **B**, **C** and **D** to distinguish the different parts of each experiment. The initial conditions **A** and **C** are linked through a *do-operator* do : $X_0 \mapsto \bar{X}_0$, which modifies the initial condition (Pearl, 2018). Experiments are parameterized by a set of intrinsic physical parameters $\boldsymbol{z}$ which are not observable from a single initial image **A**. We refer to these as *confounders*. As in *CoPhy*, in our benchmark, the do-operator is observed during training, but confounders are not, i.e. they have been used to generate the data, but are not used during training or testing. Following (Pearl, 2018), the counterfactual task consists in inferring the counterfactual outcome **D** given the observed initial condition **A**, the observed trajectory **B**, and the counterfactual initial state **C**, following a three-step process:

1. **Abduction** uses the observed data **AB** to compute the counterfactual variables, i.e. physical parameters, which are not affected by the do-operation.
2. **Action** updates the causal model; keep the same identified confounders and apply the do-operator, i.e. replace the initial state **A** by **C**.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) `BlocktowerCF` (BT-CF) (b) `BallsCF` (B-CF) (c) `CollisionCF` (C-CF)

Figure 9.1: **Dataset overview** – The Filtered-CoPhy benchmark suite contains three challenging scenarios involving 2D or 3D rigid body dynamics with complex interactions, including collision and resting contact. Initial conditions **A** are modified to **C** by an intervention. The initial motion is indicated with arrows.

3. **Prediction** computes the counterfactual outcome **D** using the causal graph.

The benchmark contains three scenarios involving rigid body dynamics. `BlocktowerCF` (BT-CF) studies stable and unstable 3D cube towers, the confounders are masses. `BallsCF` (B-CF) focuses on 2D collisions between moving spheres (confounders are masses and initial velocities). `CollisionCF` (C-CF) is about collisions between a sphere and a cylinder (confounders are masses and initial velocities, see figure 9.1).

Unlike *CoPhy*, our benchmark involves predictions in RGB pixel space only. The do-operation consists of visually observable interventions on **A**, such as moving or removing an object. The confounders cannot be identified from the single-frame observation **A**, identification requires the analysis of the entire **AB** trajectory.

For an experiment $(\mathbf{AB}, \mathbf{CD}, z)$ to be well-posed, the confounders z must be retrievable from **AB**. For example, since the masses of a stable cube tower cannot be identified generally in all situations, it can be impossible to predict the counterfactual outcome of an unstable tower, as collisions are not resolvable without known masses. In contrast to *CoPhy*, we ensure that each experiment $\psi : (X_0, z) \mapsto [X_1, \cdots, X_T]$, given initial condition X_0 and confounders z, is well-posed and satisfies the following constraints:

Definition 9.1 ***Identifiability*** *(Pearl, 2018) The experiment* $(\mathbf{AB}, \mathbf{CD}, z)$ *is identifiable if, for any set of confounders* z'*:*

$$\psi(\mathbf{A}, z) = \psi(\mathbf{A}, z') \Rightarrow \psi(\mathbf{C}, z) = \psi(\mathbf{C}, z'). \tag{9-1}$$

In an identifiable experiment, there is no distinct pair (z, z') that gives the same trajectory **AB** but different counterfactual outcomes **CD**. Details on implementation and impact are in appendix F.1.1.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

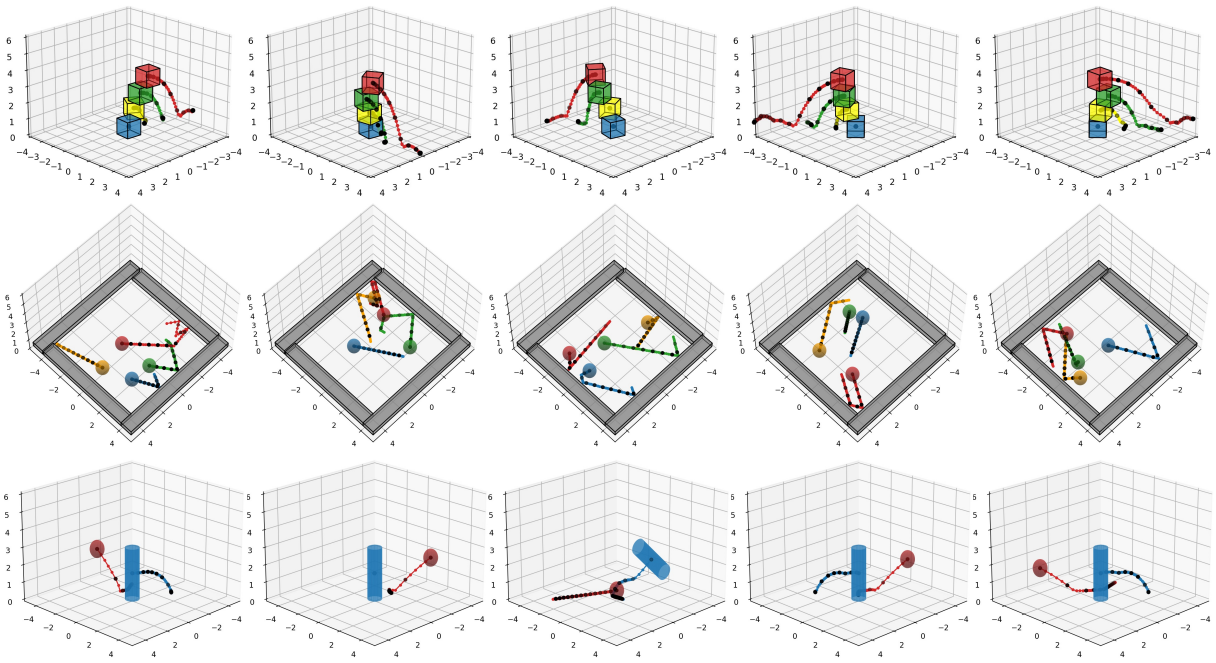


Figure 9.2: **Impact of temporal frequency** – 3D trajectories of each cube are shown. Black dots are sampled at 5 FPS, colored dots at 25 FPS. Collisions between the red cube and the ground are not well described by the black dots, making it hard to infer physical laws from regularities in data.

We also enforce sufficient difficulty of the problem through the meaningfulness of confounders. We remove initial situations where the choice of confounder values has no significant impact on the outcome:

Definition 9.2 ***Counterfactuality*** *Let z^k be the confounders $\mathbf{z}$, where the kth value has been modified. The experiment $(\mathbf{AB}, \mathbf{CD}, \mathbf{z})$ is counterfactual if and only if:*

$$\exists k : \psi(\mathbf{C}, z^k) \neq \psi(\mathbf{C}, \mathbf{z}). \tag{9-2}$$

In other words, we impose the existence of an object of the scene for which the (unobserved) physical properties have a determining effect on the trajectory. Details on how this constraint was enforced are given in appendix F.1.2.

Temporal resolution – the physical laws we target involve highly non-linear phenomena, in particular collision and resting contacts. Collisions are difficult to learn because their actions are both intense, brief, and highly non-linear, depending on the geometry of the objects in 3D space. The temporal resolution of physical simulations is of prime importance. A parallel can be made with Nyquist-Shannon frequency, as a trajectory sampled with too low frequency cannot be reconstructed with precision. We simulate and record trajectories at 25 FPS, compared to 5 FPS chosen in *CoPhy*, justified with two experiments. Firstly, figure 9.2 shows the trajectories of the center of masses of cubes in `BlocktowerCF` (and other tasks), colored dots are shown at 25 FPS and black dots at 5 FPS. We can see that collisions with the ground fall below the sampling rate of 5 FPS, making it hard to infer physical laws from regularities in

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

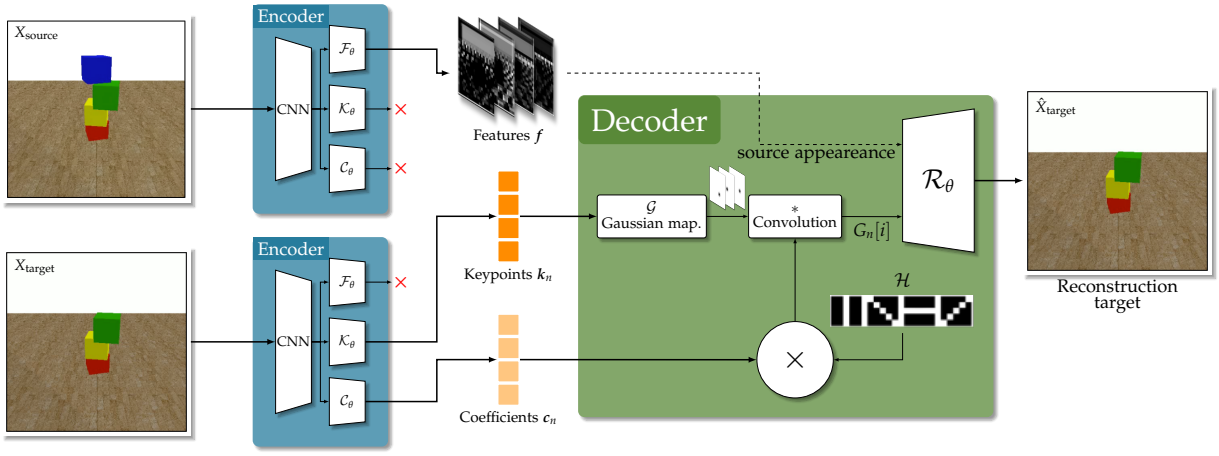


Figure 9.3: **Derendering module overview** – We de-render visual input into a latent space composed of a dense feature map f modeling static information, a set of keypoints k_n, and associated coefficients c_n. We show the training configuration taking as input pairs (X_{source}, X_{target}) of images. Without any supervision, a tracking strategy emerges naturally through the unsupervised objective: we optimize the reconstruction of X_{target}, given features from X_{source} and keypoints and coefficients from X_{target}.

data at this frequency. A second experiment involves learning a prediction model at different frequencies, confirming the choice 25 FPS — details are given in appendix F.1.3.

9.3 Unsupervised learning of counterfactual physics

We introduce a new model for counterfactual learning of physical processes capable of predicting visual sequences D in the image space over long horizons. The method does not require any supervision other than videos of observed and counterfactual experiences. The code is publicly available online at https://filteredcophy.github.io. The model consists of three parts:

1. **An image-to-keypoints encoder** learns a hybrid representation of an image in the form of a (i) dense feature map and (ii) 2D keypoints combined with (iii) a low-dimensional vector of coefficients (see figure 9.3). Without any state supervision, we show that the model learns a representation that encodes positions in keypoints and appearance and orientation in the coefficients.
2. **A Counterfactual Dynamics (CoDy) model** based on recurrent graph networks, in the lines of (Baradel et al., 2020). It estimates a latent representation of the confounders z from the keypoint and coefficient trajectories of **AB** provided by the encoder and then predicts D in this same space.
3. **A keypoints-to-image decoder** that uses the predicted keypoints to generate a pixel-space representation of D.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

9.3.1 *Disentangling visual information from dynamics*

The encoder takes an input image X and predicts a representation with three streams, sharing a common convolutional backbone, as shown in figure 9.3. Using three different heads $\mathcal{F}_\theta$, $\mathcal{K}_\theta$ and $\mathcal{C}_\theta$, the encoder computes a dense feature map $\boldsymbol{f} \in \mathbb{R}^{n_c \times n_f \times n_f}$ (n_c channels of $n_f \times n_f$ matrices) containing static information, such as the background, as well as N keypoints-coefficients pairs $(\boldsymbol{k}_n, \boldsymbol{c}_n), n \in [\![0, N]\!]$ where $\boldsymbol{k}_n \in \mathbb{R}^2$ is a 2D keypoint carrying positional information from moving objects and $\boldsymbol{c}_n \in \mathbb{R}^{C+1}$ its associated shape coefficients.

The unsupervised objective is formulated on pairs of images $(X_{\text{source}}, X_{\text{target}})$ randomly sampled from the same $\mathbf{D}$ sequences (see appendix F.4.1 for details on sampling). Exploiting an assumption on the absence of camera motion[1], the goal is to favor the emergence of disentangled static and dynamic information. To this end, both images are encoded, and the reconstruction of the target image is predicted with a decoder $\mathcal{D}_\theta$ fusing the source dense feature map and the target keypoints and coefficients. This formulation requires the decoder to aggregate dense information from the source and sparse values from the target, naturally leading to motion being predicted by the latter.

The decoder $\mathcal{D}_\theta$ uses inductive bias favoring the usage of the 2D keypoint information in a spatial way. The 2D coordinates $\boldsymbol{k}_n$ for each keypoint are encoded as Gaussian heatmaps $\mathcal{G}(\boldsymbol{k}_n)$, i.e. 2D Gaussian functions centered on the keypoint position. The additional coefficient information, carrying appearance information, is then used to deform the Gaussian mapping into an anisotropic shape using a fixed filter bank $\mathcal{H} = \{H_1, \cdots, H_C\}$, such as

$$\mathcal{D}_\theta\Big(\boldsymbol{f}, \boldsymbol{k}_1, \boldsymbol{c}_1, \cdots, \boldsymbol{k}_N, \boldsymbol{c}_N\Big) = \mathcal{R}_\theta\left(\boldsymbol{f}, \overbrace{\begin{bmatrix} G_1[1] \\ \vdots \\ G_1[C], \end{bmatrix}}^{\text{mapping for keypoint 1}}, \cdots, \overbrace{\begin{bmatrix} G_N[1] \\ \vdots \\ G_N[C], \end{bmatrix}}^{\text{mapping for keypoint n}}\right)$$

$$\text{where } G_n[i] = \boldsymbol{c}_n^{C+1}(\underbrace{\boldsymbol{c}_n^i H_i}_{\text{Filter } i}) * \mathcal{G}(\boldsymbol{k}_n), \quad (9\text{-}3)$$

where $\mathcal{R}_\theta$ is a refinement network performing trained upsampling with transposed convolutions, whose inputs are stacked channelwise. $G_n[i]$ are Gaussian mappings produced from keypoint positions $\boldsymbol{k}_n$, deformed by filters from bank $\mathcal{H}$ and weighted by coefficients $\boldsymbol{c}_n^i$. The filters H_i are defined as fixed horizontal, vertical, and diagonal convolution kernels. This choice is discussed in section 9.4. The joint encoding and decoding pipeline is illustrated in Fig. 9.3.

The model is trained to minimize the mean squared error (MSE) reconstruction loss, regularized with a loss on spatial gradients ∇X weighted by hyper-parameters $\gamma_1, \gamma_2 \in \mathbb{R}$:

$$\mathcal{L}_{\text{deren}} = \gamma_1 |X_{\text{target}} - \hat{X}_{\text{target}}|^2 + \gamma_2 |\nabla X_{\text{target}} - \nabla \hat{X}_{\text{target}}|^2, \quad (9\text{-}4)$$

[1] If this assumption is not satisfied, global camera motion could be compensated after estimation.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

where $\hat{X}_{\text{target}}=\mathcal{D}_\theta(\mathcal{F}(X_{\text{source}}),\mathcal{K}_\theta(X_{\text{target}}),\mathcal{C}_\theta(X_{\text{target}}))$ is the reconstructed image.

Related work – our unsupervised objective is somewhat related to the *Transporter* (Kulkarni et al., 2019), which, computes visual feature vectors $\boldsymbol{f}_{\text{source}}$ and $\boldsymbol{f}_{\text{target}}$ as well as 2D keypoints $\boldsymbol{k}_n^{\text{source}}$ and $\boldsymbol{k}_n^{\text{target}}$, modeled as a 2D vector via Gaussian mapping. It leverages a handcrafted transport equation: $\hat{\Psi}_n^{\text{target}}=\boldsymbol{f}_{\text{source}}\times(1-\boldsymbol{k}_n^{\text{source}})\times(1-\boldsymbol{k}_n^{\text{target}})+\boldsymbol{f}_{\text{target}}\times\boldsymbol{k}_n^{\text{target}}$. Similarly, the target image is reconstructed through a refiner network $\hat{X}_{\text{target}}=\mathcal{R}_\theta(\hat{\Psi}_1^{\text{target}},\cdots,\hat{\Psi}_N^{\text{target}})$. Yet, the transporter suffers from a major drawback when used for video prediction, as it requires parts of the target image to reconstruct the target image (i.e. $\boldsymbol{f}_{\text{target}}$). The model was originally proposed in the context of RL and control, where reconstruction is not an objective. It also does not use shape coefficients, requiring shapes to be encoded by several keypoints, or abusively be carried through the dense features $\boldsymbol{f}_{\text{target}}$. This typically leads to complex dynamics nonrepresentative of the dynamical objects. We conducted an in-depth comparison between the Transporter and our representation in appendix F.3.2.

9.3.2 *Dynamic model and confounder estimation*

Our counterfactual dynamical model (CoDy) leverages multiple GNNs modules (Battaglia et al., 2016) that join forces to solve the counterfactual forecasting tasks of Filtered-CoPhy. Each one of these networks is a classical GNN, abbreviated as $f_{\text{gnn}}(\boldsymbol{x}_k)$, which contextualizes input node embeddings $\boldsymbol{x}_k$ through incoming edge interactions $\boldsymbol{e}_{ik}$, providing output node embeddings $\hat{\boldsymbol{x}}_k$ (parameters are not shared over the instances)

$$f_{\text{gnn}}(\boldsymbol{x}_k)=\hat{\boldsymbol{x}}_k,\ \text{such that}\ \hat{\boldsymbol{x}}_k=g_{\text{nodes}}\left(\boldsymbol{x}_k,\sum_i \boldsymbol{e}_{ik}\right)\ \text{with}\ \boldsymbol{e}_{ij}=g_{\text{edges}}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right), \tag{9-5}$$

where g_{edges} is a message-passing function and g_{nodes} is an aggregation function.

We define the state of a frame X_t at time t as a stacked vector $\boldsymbol{s}[t]$ composed of keypoints and coefficients computed by the image-to-keypoints encoder, i.e.

$$\boldsymbol{s}[t]=\begin{bmatrix}\boldsymbol{s}_1[t] & \cdots & \boldsymbol{s}_k[t]\end{bmatrix}\ \text{where}\ \boldsymbol{s}_n[t]=\begin{bmatrix}\boldsymbol{k}_n & \boldsymbol{c}_n\end{bmatrix}[t]. \tag{9-6}$$

In the lines of (Baradel et al., 2020), given the original initial condition and outcome **AB**, CoDy estimates an unsupervised representation $\tilde{\boldsymbol{z}}_n$ of the latent confounder variables per keypoint n through the counterfactual estimator (*CF estimator* in figure 9.4). It first contextualizes the sequence $\boldsymbol{s}^{\mathbf{AB}}[t]$ through a GNN, then model the temporal evolution of this representation with a GRU (Cho et al., 2014) per keypoint, sharing parameters over keypoints. Formally

$$\boldsymbol{h}^{\mathbf{AB}}[0]=0,\quad \boldsymbol{h}^{\mathbf{AB}}[t+1]=r_{\text{gru},1}\left(\boldsymbol{h}^{\mathbf{AB}}[t],f_{\text{gnn},1}\left(\boldsymbol{s}^{\mathbf{AB}}[t]\right)\right),\quad \tilde{\boldsymbol{z}}=\boldsymbol{h}^{\mathbf{AB}}[T] \tag{9-7}$$

We took inspiration from chapter 5, showing, under mild assumptions, the existence of a latent space of higher dimension, where a dynamical system given as an ODE can have simpler dynamics. We thus propose to use an encoder-decoder structure within CoDy, which

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

projects our dynamical system into a higher-dimensional state space, performs forecasting of the dynamics in this latent space, and then projects predictions back to the original keypoint space. Note that this dynamics encoder/decoder is different from the encoder/decoder of the image processing modules discussed in section 9.3.1, and resembles the Latent Dynamics discussed in chapter 2.

The state encoder $\mathcal{E}(\boldsymbol{s}[t]) = \sigma[t]$ is modeled as a graph network $\mathcal{E} := f_{2,\text{gnn}}$, whose aggregation function projects into an output embedding space $\sigma[t]$ of dimension 256. The decoder $\Delta(\sigma[t]) = \boldsymbol{s}[t]$ temporally processes the individual contextualized states $\sigma[t]$ with a GRU, followed by new contextualization with a graph network $f_{3,\text{gnn}}$. Details on the full architecture are provided in appendix F.4.2.

The dynamic model CoDy performs forecasting in the higher-dimensional space $\sigma[t]$, computing a displacement vector $\delta[t+1]$ such that $\sigma[t+1] = \sigma[t] + \delta[t+1]$. It takes the projected state embeddings $\sigma_n[t]$ per keypoint n concatenated with the confounder representation $\tilde{\boldsymbol{z}}_n$ and contextualizes them with a GNN, resulting in embeddings $\boldsymbol{u}_n[t] = f_{4,\text{ gnn}}\Big([\sigma_n[t], \tilde{\boldsymbol{z}}_n]\Big)$, which are processed temporally by a GRU. We compute the displacement vector at time $t+1$ as a linear transformation from the hidden state of the GRU. We apply the dynamic model in an auto-regressive way to forecast long-term trajectories in the projected space $\sigma_n[t]$ and apply the state decoder to obtain a prediction $\hat{\boldsymbol{s}}^{\text{CD}}[t]$. The dynamic model is trained with a loss in keypoint space,

$$\mathcal{L}_{\text{physics}} = \sum_t \left|\boldsymbol{s}^{\text{CD}}[t] - \hat{\boldsymbol{s}}^{\text{CD}}[t]\right|^2 + \gamma_3 \left|\boldsymbol{s}^{\text{CD}}[t] - \Delta\Big(\mathcal{E}\left(\boldsymbol{s}^{\text{CD}}[t]\right)\Big)\right|^2. \tag{9-8}$$

The first term enforces the model to learn to predict the outcomes and the second term favors correct reconstruction of the state in keypoint space. The terms are weighted with a scalar parameter γ_3.

9.3.3 *Training*

End-to-end training of all three modules jointly is challenging, as the same pipeline controls both the keypoint-based state representation and the dynamic module (CoDy), involving two adversarial objectives: optimizing reconstruction pushes the keypoint encoding to be as representative as possible, but learning the dynamics favors a simple representation. Faced with these two contradictory tasks, the model is numerically unstable and rapidly converges to regression to the mean. As described above, we separately train the encoder and decoder pair on reconstruction only, without dynamical information (see equation (9-4)). Then we freeze the parameters of the keypoint detector and train CoDy to forecast the keypoints from D minimizing the loss in Equation (9-8).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

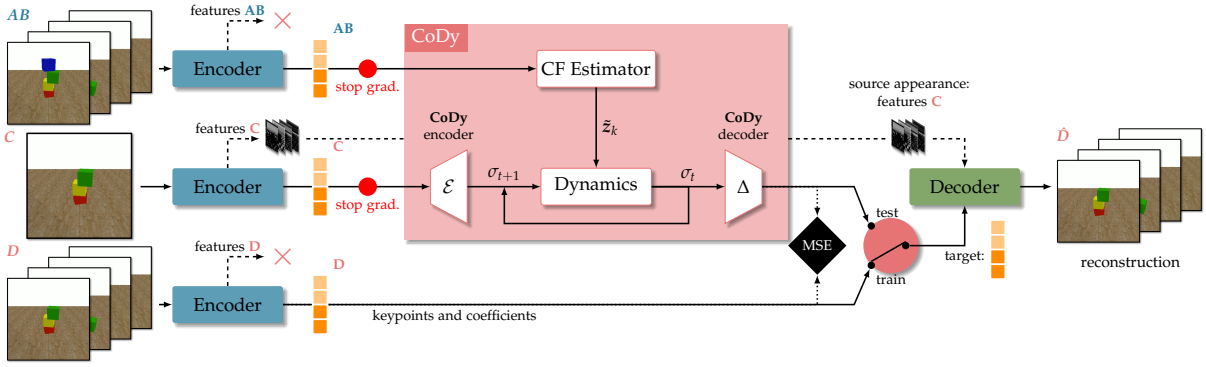


Figure 9.4: **Model overview** – During training, we disconnect the dynamic prediction module (CoDy) from the rendering module (decoder). On test time, we reconnect the two modules. CoDy forecasts the counterfactual outcome D from the sparse keypoints representation of AB and C. The confounders are discovered in an unsupervised manner and provided to the dynamical model.

		Ours		UV-CDN		PhyD	Pred
		K	2K	K	2K	NET	RNN
BT-CF	PSNR	23.48	**24.69**	21.07	21.99	16.49	22.04
	L-PSNR	25.51	**26.79**	22.36	23.64	23.03	24.97
B-CF	PSNR	21.19	**21.33**	19.51	19.54	18.56	22.31
	L-PSNR	**23.88**	24.12	22.35	22.38	22.55	22.63
C-CF	PSNR	24.09	24.09	23.73	23.83	19.69	**24.70**
	L-PSNR	26.07	**26.55**	26.08	26.34	24.61	26.39

Table 9.1: **Quantitative results** – Comparison with the state-of-the-art models in physics-related machine learning of video signals, reporting reconstruction error (PSNR and introduced L-PSNR).

	Copy B	Copy C	**Ours**
BT-CF	43.2	20.0	**9.58**
B-CF	44.3	92.3	**36.12**
C-CF	7.6	40.3	**5.14**

Copy B: absence of intervention (always outputs **B**).
Copy C: the tower is stable (always outputs **C**).

Table 9.2: **Copying baselines** – We report MSE$\times 10^{-3}$ on prediction of keypoints and coefficients.

9.4 Experiments

We compare the proposed model to three strong baselines for physics-inspired video prediction.

- **PhyDNet** (Guen and Thome, 2020) is a non-counterfactual video prediction model that forecasts future frames using a decomposition between (a) a feature vector that temporally evolves via an LSTM and (b) a dynamic state that follows a PDE learned through specifically designed cells (see chapter 3 for an overview).
- **V-CDN** (Li et al., 2020a) is a counterfactual model based on keypoints, close to our work. It identifies confounders from the beginning of a sequence and learns a keypoint predictor through auto-encoding using the Transporter equation (see discussion in section 9.3.1). As it stands, it cannot be used for video prediction and is incomparable with our work, see details in appendix F.3. We therefore replace the Transporter with our image encoder-decoder modules, from which we remove the additional coefficients. We refer

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

# Coefficients:		✓			✗		
# Keypoints:		K	2K	4K	K	2K	4K
BT-CF	PSNR	23.48	**24.69**	23.54	22.71	23.28	23.17
	L-PSNR	21.75	**23.03**	21.80	21.18	21.86	21.70
B-CF	PSNR	21.19	21.33	**21.37**	20.49	21.09	20.97
	L-PSNR	**27.88**	27.16	27.07	26.33	27.07	26.73
C-CF	PSNR	24.09	24.09	**24.26**	23.84	23.66	24.06
	L-PSNR	23.32	**23.46**	23.44	22.58	22.81	23.45

Table 9.3: **Ablation on coefficients** – Impact of having additional orientation/shape coefficients (✓) compared to the keypoint-only solution (✗), for different numbers of keypoints: equal to number of objects (= K), $2K$ and $4K$.

to this model as UV-CDN (for Unsupervised V-CDN).

- **PredRNN** (Wang et al., 2017) is a ConvLSTM-based video prediction model that leverages spatial and temporal memories through a spatiotemporal LSTM cell.

All models have been implemented in PyTorch, architectures are described in appendix F.4. For the baselines PhyDNet, UV-CDN, and PredRNN, we used the official source code provided by the authors. We evaluate on each scenario of Filtered-CoPhy on the counterfactual video prediction task. For the two counterfactual models (Ours and UV-CDN), we evaluate on the tasks as intended: we provide the observed sequence **AB** and the CF initial condition C, and forecast the sequence D. The non-CF baselines are required to predict the entire video from a single frame to prevent them from leveraging shortcuts in a part of the video and bypass the need for physical reasoning.

We measure performance with time-averaged peak signal-to-noise ratio (PSNR) that directly measures reconstruction quality. However, this metric is mainly dominated by errors on the static background, which is not our main interest. We also introduce Localized PSNR (L-PSNR), which measures area error in the important regions near moving objects, computed on masked images. We compute the masks using classical background subtraction techniques.

Comparison to the SOTA – We compare our model against UV-CDN, PhyDNet, and PredRNN in Table 9.1, consistently and significantly outperforming the baselines. The gap with UV-CDN is particularly interesting, as it confirms the choice of additional coefficients to model the dynamics of moving objects. PredRNN shows competitive performances, especially on `CollisionCF`. However, our localized PSNR tends to indicate that the baseline does not reconstruct accurately the foreground, favoring the reconstruction of the background to the detriment of the dynamics of the scene. Fig. 9.5 visualizes the prediction on a single example, more can be found in appendix F.7. We also compare to trivial copying baselines in Table 9.2, namely *Copy B*, which assumes no intervention and outputs the **B** sequence, and *Copy C*, which assumes a stable tower. We evaluate these models in keypoints space measuring MSE on keypoints and coefficients averaged over time, as copying baselines are unbeatable in the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

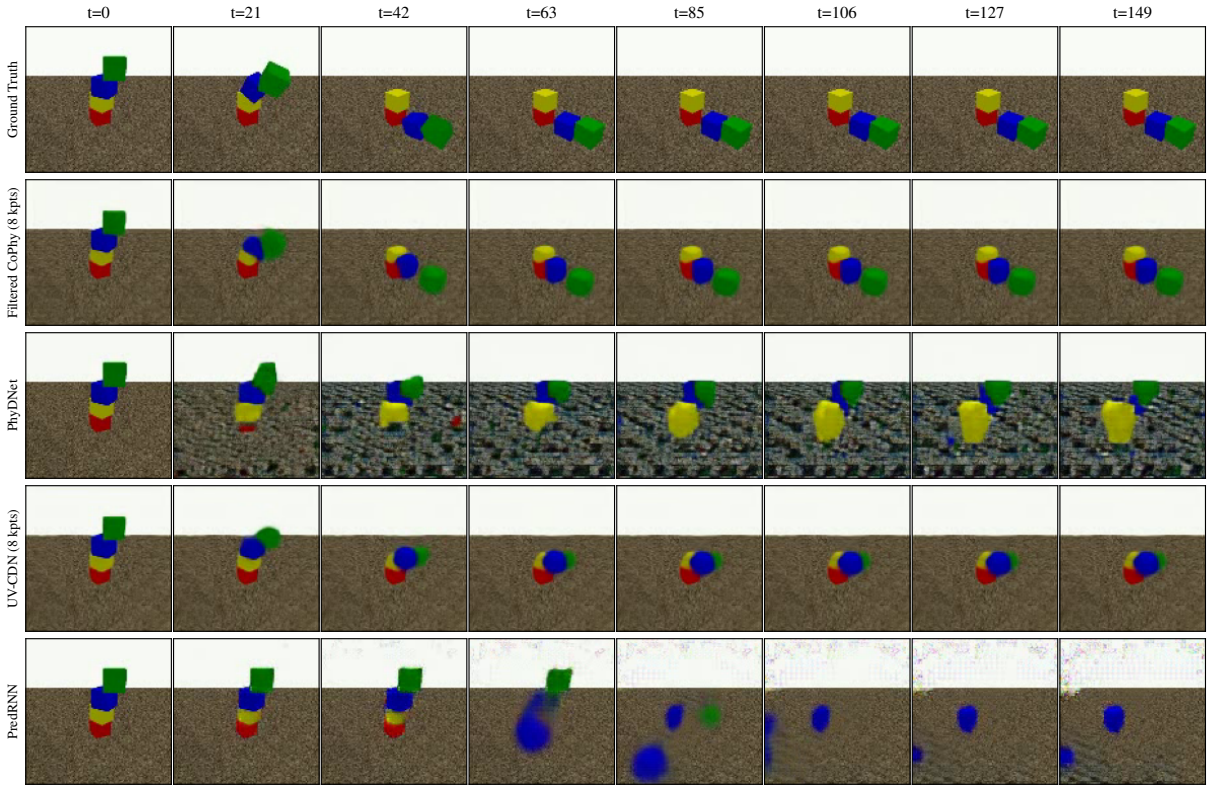


Figure 9.5: **Qualitative results** – Visualization of the counterfactual video prediction quality, comparing our proposed model (Filtered-CoPhy) with the two baselines, PhyDNet and UV-CDN, over different timesteps.

regions of static background, making the PSNR metrics unusable.

We provide additional empirical results by comparing the models using Multi-object Tracking metrics and studies on the impact of the do-operations on PSNR in appendix F.5. We also compute an upper bound of our model using the CoPhyNet baseline as described in Baradel et al. (2020).

Performance on real-world data – is reported in appendix F.6, showing experiments on 516 videos of real wooden blocks introduced in Lerer et al. (2016).

Impact of appearance coefficients – are reported in Table 9.3, comparing to the baseline using a keypoint-only representation. The coefficients have a significant impact: even increasing the number of keypoints to compensate for the loss of information cannot overcome the advantage of disentangling positions and shapes, as done in our model. We provide a deeper analysis of the de-rendering/rendering modules in appendix F.2, which includes visualizations of the navigation of the latent shape space in F.2.2.

Learning filters – does not have a positive impact on reconstruction performance compared to the choice of the handcrafted bank, as can be seen in table 9.5. We conjecture that the additional degrees of freedom are redundant with parts of the filter kernels in the refinement

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

State auto-encoder:	✓	✗
BT-CF	**9.58**	11.10
B-CF	**36.12**	36.88
C-CF	**5.14**	16.16

Table 9.4: **Ablation on CoDy** – Impact of the dynamical CoDy encoder (✓) against the baseline operating in the keypoint and coefficient space (✗). We report MSE $\times 10^{-3}$ on the prediction of keypoints and coefficients (4 keypoints).

Filter bank:	Fixed	Learned
BT-CF	**34.40**	32.04
B-CF	**37.76**	31.25
C-CF	**34.09**	33.88

Table 9.5: **Ablation of the filters** – Learning the filter bank $\mathcal{H}$ from scratch has a mild negative effect on the reconstruction task. We report the PSNR on static reconstruction performance without the dynamical model.

module $\mathcal{R}_\theta$: this corresponds to jointly learning a multi-channel representation $\{G_n[i] \,|\, i \in [\![1, C]\!], n \in [\![1, N]\!]\}$ for shapes as well as the mapping which geometrically distorts them into the target object shapes. Fixing the latent representation does not constrain the system, as the mapping $\mathcal{R}_\theta$ can adjust to it.

Impact of the high-dimensional dynamic space – We evaluate the impact of modeling object dynamics in high-dimensional space through the CoDy encoder in Table 9.4, comparing projection to 256 dimensions to the baseline reasoning directly in keypoint and coefficient space. The experiment confirms this choice of KKL-like encoder.

9.5 Conclusion

We introduced a new benchmark for counterfactual reasoning in physical processes requiring to perform video prediction, i.e. predicting raw pixel observations over a long horizon. The benchmark has been carefully designed and generated imposing constraints on identifiability and counterfactuality. We also propose a new method for counterfactual reasoning, which is based on a hybrid latent representation combining 2D keypoints and additional latent vectors encoding appearance and shape. We introduce an unsupervised learning algorithm for this representation, which does not require any supervision on confounders or other object properties and processes raw video. Counterfactual prediction of video frames remains a challenging task, and Filtered-CoPhy still exhibits failures in maintaining rigid structures of objects over long prediction horizons.

9.6 Post-Scriptum: taking a step back

At first glance, Filtered-CoPhy seems like an easy task. Two years ago (when this project was released), the literature was full of research on computer vision that could be potentially used to process the videos. Moreover, how difficult could it be to simulate a simple tower of cubes? Underestimating the difficulty of Filtered-CoPhy was arguably the first mistake of this Ph.D.

About rigid body dynamics – the physics in Filtered-CoPhy is actually very difficult to simulate using neural networks. There are two potential sources of difficulties. Firstly, collisions

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

between rigid bodies imply brutal changes in momentum, in physics, we talk about *impulses* rather than forces. These impulses strongly depend on the geometry of the objects, their velocities, and their physical parameters. This type of highly non-linear change of dynamics is very difficult to approximate using neural networks, which tend to model Lipschitz functions. Consider `Balls-CF` for instance: most of the time, the dynamics is fairly simple, i.e. linear translation, but drastically changes on a handful of frames when two objects collide. One could think of using two models, one for translation (a simple linear dynamics is enough), and a more powerful neural network handling collision. This implies the use of a third collision detector model to control the switch between both dynamics, which is difficult to train without access to the ground truth trajectory.

Such an approach could be promising for `Balls-CF` and, up to some extent, for `Collision-CF`, but neglect another key difficulty: resting contacts. This phenomenon occurs when multiple objects are in contact. The impulses of the objects are then interconnected, and simulating the correct trajectory involves solving a very difficult optimization problem. Even when ground truth trajectories are available, rigid body dynamics is difficult to model. For instance Allen et al. (2022b) propose a face interaction network (a GNN handling message passing at node, edges, and faces level) to simulate rigid bodies, and while showing promising results, still provides inaccurate predictions, especially on collisions. In comparison, Filtered-CoPhy is even more challenging, since it has a counterfactual aspect and does not allow for ground truth supervision.

Model improvements – concerning the image encoder-decoder structure, our design can surely benefit from several improvements. We could probably head towards vision transformers (Dosovitskiy et al., 2021) to replace the CNN encoder since this type of architecture has shown great success during the last two years. Additionally, the keypoint detection mechanism could be replaced with an attention mechanism at pixel level (or even below). The decoder could also be updated using diffusion models (Rombach et al., 2022) from the feature maps and conditioned on the keypoints and coefficients. While our structure already discover how to encode (up to some extent) shape information in the coefficients, the use of diffusion models could help produce crisper images with cleaner contours.

CoDy is more difficult to improve. The fact that learning the dynamics is carried by unsupervised keypoints limits the possibilities (for instance, face interaction network Allen et al. (2022b) cannot be used readily). A possibility is to replace the dynamics with a differentiable simulator and train the model end-to-end. This forces the keypoints to match physical coordinates, and coefficients to correspond to a standard rotation (such as quaternions or rotation matrices). The confounders can be retrieved with an inner optimization loop trying to reproduce the trajectory **AB**, and then reused to simulate D^2.

[2] All these research tracks are conditioned to the existence of another suitable coffee-related pun, such as Latte-CoPhy or Cold-CoPhy.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Counterfactual learning – one of the limitations of our method is that it requires both the observed *and* the modified experiment for training. A proper counterfactual reasoning task (following Pearl (2000)) should not assume knowledge of the do-operation during training, while still being able to generalize to counterfactual experiments during test. A simple modification to our model consists of training the dynamics on **B** rather than D, hence using the same trajectory for confounders estimation and physics forecasting. In practice, preventing overfitting is difficult: a lot of information can flow from the CF estimator to the dynamics via the latent vector $\tilde{z}$. Doing so, CoDy overfits and fails to extract the confounders from **AB** but rather encodes the entire trajectory in the latent vector. Thus, to respect strictly the counterfactual task, we must add prior knowledge about the system (i.e. enforce known causal relationships), by using for instance a differentiable simulator. Another option is to search for minimal modification of the causal graph to integrate the effect of the do-operation.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chapter 10

Final remarks

We have systematically broken down perspectives on a chapter-by-chapter basis, pinpointing potential shortcomings of our work and tracks for improvements. When possible, we proposed insights for addressing these issues, and we hope that our contributions will raise interest from the community and motivate interesting and effective follow-up work in the field of hybrid physics and research in deep learning. Most work discussed in this manuscript has led to potential new collaborations with our colleagues, yet to be explored, especially in the field of control theory. On this conclusive note, we would like to draw a bigger picture and propose perspectives on a larger scale for the domain of hybrid physics and deep learning techniques.

10.1 Theoretical insights for more principled models

In our opinion, the main difference between machine learning approaches for physics and conventional tools is that research in the field advances faster than theoretical results. Deep learning addresses harder and harder tasks, while theoretical results on their performance and robustness are late to come. Indeed, on one hand, provable convergence guarantees and error bounds are extremely difficult to obtain with the convoluted neural architectures used in modern approaches, and dedicating time to this subject can meet with a sense of frustration. On the other hand, because very few theoretical tools exist, research in neural networks often boils down to empirical results on tailored and curated datasets, with limited (or at least untested) portability to the real-world. To some extent, our work is no exception: models introduced in Part IV of this manuscript rely on insights from physics, prior knowledge, and reasonable intuitions, but no formal results. Conversely, techniques from Part II are better supported by theory but are difficult to scale to larger and harder tasks.

We argue for a balance between analytical and empirical results. We believe that building upon properties from control theory to derive more advanced architectures is a promising way to obtain new models and techniques to improve the accuracy and robustness of physics-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

oriented learning. Yet, in most cases, the assumptions and requirements for applying these theorems cannot be rigorously verified, and the practical implementation might slightly differ from the analytical framework. Yet, we believe it is beneficial to take inspiration from these results to influence our design and implementation processes. Many areas of research in control theory are promising for applications of deep learning. We explored contraction theory, which allows to give insights into the convergence and stability of recurrent neural networks, and argue for adopting these principles more broadly to construct architectures that converge to a meaningful solution more robustly. Moreover, non-linear control theory, observer/controller design, graph theory, and cooperative game theory are potentially promising fields to be explored and fused with deep learning to contribute to the development of more principled and interpretable models.

10.2 Physics and deep learning for robotics

We observed that most advanced deep learning-based approaches for physics address autonomous systems, i.e. with input signals. This is also the case of our work: among the previous six chapters, only two address non-autonomous dynamical systems (chapter 4 and chapter 6). Indeed, taking into account input signals is not trivial since they significantly complicate the dynamics and require a more comprehensive understanding of the system, which results in larger datasets. Yet, we believe that robotics is among the most promising applications of deep learning, but this field implies interacting with an environment, hence the presence of input control. Nowadays, most research directions replace conventional control algorithms with reinforcement learning setups. These approaches are promising and successful in a growing number of tasks, but we believe that the domain can benefit from more hybridization, i.e. combining deep learning techniques with physical models. This can help in designing architectures that are more intuitive, easy to understand and implement, and potentially more efficient.

Using physics priors in robotics may also be a key to accelerating the release of our models in the outside world. Relying on control and physics insights might help offer security and robustness guarantees that are necessary for real-world applications. By fusing conventional tools and modern learning approaches, we can design models that inherently respect the laws of nature, leading to more robust and safer outcomes.

10.3 Neural simulators for faster engineering

In the introduction of the manuscript, we stated that most common phenomena in physics can already be simulated using classical tools, such as standard PDE and ODE solvers. These methods are fairly accurate and provide stability guarantees. We argue that research in neural simulators serves two purposes. The main interest is obviously computation speed since neural networks can rely on intuitive physics rather than handcrafted equations to produce

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

long-term simulations in a matter of seconds. When working on a new design, simulation is a way to gain feedback on how a system will behave in the real world, hence offering wider and faster access to simulations increases the amount of feedback that engineers can obtain, allowing them to discover better designs more quickly.

We strongly believe that the long-term goal of neural simulators relies upon the differentiability of the solver, which can thus be used to optimize a process according to a well-crafted objective directly using the solver (with gradient descent), which traditional simulation tools cannot handle well. This opens the possibility of using neural simulators for the automatic design and optimization of systems. For instance, one could train a model to learn the dynamics of a physical system, and then use this model in an optimization loop to find the best design parameters for the system. This approach can drastically speed up the design and optimization process, particularly for complex systems where there are many parameters to adjust. The ability to automatically tune and optimize parameters based on simulation results improves the design process and results in better-performing systems.

We hope that these perspectives will inspire more researchers to explore the intersection between physics, control theory, and deep learning. By bringing together knowledge from these distinct but interrelated fields, we can push the boundaries of what is currently possible and open up exciting new avenues of research.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Bibliography

Abate, A., Ahmed, D., Giacobbe, M., and Peruffo, A. (2020). Formal synthesis of lyapunov neural networks. *Control Systems Letters*.

Ajay, A., Wu, J., Fazeli, N., Bauza, M., Kaelbling, L. P., Tenenbaum, J. B., and Rodriguez, A. (2018). Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *International Conference on Intelligent Robots and Systems*.

Allen, K., Lopez-Guevara, T., Stachenfeld, K. L., Gonzalez, A. S., Battaglia, P., Hamrick, J. B., and Pfaff, T. (2022a). Inverse design for fluid-structure interactions using graph network simulators. In *Neural Information Processing Systems*.

Allen, K. R., Rubanova, Y., Lopez-Guevara, T., Whitney, W. F., Sanchez-Gonzalez, A., Battaglia, P., and Pfaff, T. (2022b). Learning rigid dynamics with face interaction graph networks. In *International Conference on Learning Representations*.

Amos, B., Xu, L., and Kolter, J. Z. (2017). Input convex neural networks. In *International Conference on Machine Learning*.

Andrieu, V. (2014). Convergence speed of nonlinear luenberger observers. *Journal on Control and Optimization*.

Andrieu, V., Jayawardhana, B., and Praly, L. (2016). Transverse exponential stability and applications. *Transactions on Automatic Control*.

Andrieu, V. and Praly, L. (2006). On the Existence of a Kazantzis-Kravaris/Luenberger Observer. *Journal on Control and Optimization*.

Antonini, A., Guerra, W., Murali, V., Sayre-McCord, T., and Karaman, S. (2018). Blackbird Dataset: A large-scale dataset for UAV perception in aggressive flight. *Robotics Research*.

Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., and Girshick, R. (2019). Phyre: A new benchmark for physical reasoning. *Neural Information Processing Systems*.

Balke, A. and Pearl, J. (1994). Counterfactual probabilities: Computational methods, bounds and applications. In *Uncertainty in Artificial Intelligence*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Baradel, F., Neverova, N., Mille, J., Mori, G., and Wolf, C. (2020). Cophy: Counterfactual learning of physical dynamics. In *International Conference on Learning Representations*.

Baradel, F., Neverova, N., Wolf, C., Mille, J., and Mori, G. (2018). Object level visual reasoning in videos. In *European Conference on Computer Vision (ECCV)*.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In *Neural Information Processing Systems*.

Barton, I. E. (1998). Comparison of simple-and piso-type algorithms for transient flows. *International Journal for Numerical Methods in Fluids*.

Battaglia, P., Hamrick, J., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv Preprint*.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. (2016). Interaction networks for learning about objects, relations and physics. *Neural Information Processing Systems*.

Bauersfeld, L., Kaufmann, E., Foehn, P., Sun, S., and Scaramuzza, D. (2021). Neurobem: Hybrid aerodynamic quadrotor model. *arXiv preprint*.

Beeching, E., Dibangoye, J., Simonin, O., and Wolf, C. (2020). Learning to plan with uncertain topological maps. In *European Conference on Computer Vision*.

Beintema, G., Toth, R., and Schoukens, M. (2021). Nonlinear state-space identification using deep encoder networks. In *Learning for Dynamics and Control*.

Beintema, G. I., Schoukens, M., and Tóth, R. (2023). Deep subspace encoders for nonlinear system identification. *Automatica*.

Belbute-Peres, F. D. A., Economon, T., and Kolter, Z. (2020). Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *International Conference on Machine Learning*.

Bemporad, A. (2006). Model predictive control design: New trends and tools. In *Conference on Decision and Control*.

Berg, J. and Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*.

Bernard, P., Andrieu, V., and Astolfi, D. (2022). Observer design for continuous-time dynamical systems. *Annual Reviews in Control*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Bernard, P. and Maghenem, M. (2023). Reconstructing indistinguishable solutions via set-valued kkl observer. *arXiv preprint*.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *Journal on Image and Video Processing*.

Bézenac, E. D., Pajot, A., and Gallinari, P. (2019). Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*.

Bhat, U. and Munch, S. B. (2022). Recurrent neural networks for partially observed dynamical systems. *Physical Review*.

Bianchi, F. M., Grattarola, D., and Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*.

Bishnoi, S., Bhattoo, R., Jayadeva, J., Ranu, S., and Krishnan, N. A. (2023). Enhancing the inductive biases of graph neural ode for modeling physical systems. In *International Conference on Learning Representations*.

Boussif, O., Bengio, Y., Benabbou, L., and Assouline, D. (2022). Magnet: Mesh agnostic neural pde solver. In *Neural Information Processing Systems*.

Breiman, L. (1996). Bagging predictors. *Machine Learning*.

Brivadis, L., Andrieu, V., Bernard, P., and Serres, U. (2023). Further remarks on kkl observers. *Systems & Control Letters*.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint*.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *National Academy of Sciences*.

Budišić, M., Mohr, R., and Mezić, I. (2012). Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*.

Buisson-Fenet, M., Bahr, L., Morgenthaler, V., and Meglio, F. D. (2022). Towards gain tuning for numerical kkl observers. *arXiv preprint*.

Buşoniu, L., de Bruin, T., Tolić, D., Kober, J., and Palunko, I. (2018). Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*.

Byravan, A. and Fox, D. (2017). Se3-nets: Learning rigid body motion using deep neural networks. In *International Conference on Robotics and Automation*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Byrnes, C. and Isidori, A. (2003). Limit sets, zero dynamics, and internal models in the problem of nonlinear output regulation. *Transaction on Automatic Control*.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. (2021a). Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. (2021b). Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*.

Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al. (2020). Spectral temporal graph neural network for multivariate time-series forecasting. In *Neural Information Processing Systems*.

Cao, Y., Chai, M., Li, M., and Jiang, C. (2022). Bi-stride multi-scale graph neural network for mesh-based physical simulation. *arXiv preprint*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European Conference on Computer Vision*.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. (2021). Grand: Graph neural diffusion. In *International Conference on Machine Learning*.

Chang, B., Chen, M., Haber, E., and Chi, E. H. (2019a). Antisymmetricrnn: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*.

Chang, Y.-C., Roohi, N., and Gao, S. (2019b). Neural lyapunov control. *Neural Information Processing Systems*.

Cheah, C.-C., Liu, C., and Slotine, J. (2006). Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models. *Transactions on Automatic Control*.

Chen, C., Xie, L., Xie, K., Lewis, F., and Xie, S. (2022). Adaptive optimal output tracking of continuous-time systems via output-feedback-based reinforcement learning. *Automatica*.

Chen, J., Hachem, E., and Viquerat, J. (2021a). Graph neural networks for laminar flow prediction around random two-dimensional shapes. *Physics of Fluids*.

Chen, K., Vicente, J. P. D., Sepulveda, G., Xia, F., Soto, A., Vázquez, M., and Savarese, S. (2019). A behavioral approach to visual navigation with graph localization networks. *arXiv preprint*.

Chen, R. T., Amos, B., and Nickel, M. (2020). Learning neural event functions for ordinary differential equations. In *International Conference on Learning Representations*.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Neural Information Processing Systems*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Chen, S., Sammak, S., Givi, P., Yurko, J. P., and Jia, X. (2021b). Reconstructing high-resolution turbulent flows using physics-guided neural networks. In *International Conference on Big Data*.

Chen, S., Xu, Y., Yu, C., Li, L., Ma, X., Xu, Z., and Hsu, D. (2023). Daxbench: Benchmarking deformable object manipulation with differentiable physics. In *International Conference on Learning Representations*.

Chen, Y., Liu, S., and Wang, X. (2021c). Learning continuous image representation with local implicit image function. In *Conference on Computer Vision and Pattern Recognition*.

Chen, Z., Liu, Y., and Sun, H. (2021d). Physics-informed learning of governing equations from scarce data. *Nature Communications*.

Cheng, D., Yang, F., Xiang, S., and Liu, J. (2022). Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*.

Cho, K., Merriënboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*.

Choudhary, A., Lindner, J. F., Holliday, E. G., Miller, S. T., Sinha, S., and Ditto, W. L. (2021). Forecasting hamiltonian dynamics without canonical coordinates. *Nonlinear Dynamics*.

Chow, T. W. and Fang, Y. (1998). A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics. *Transactions on Industrial Electronics*.

Cioffi, G., Bauersfeld, L., Kaufmann, E., and Scaramuzza, D. (2023). Learned inertial odometry for autonomous drone racing. *Robotics and Automation Letters*.

Cook, R. L. (1986). Stochastic sampling in computer graphics. *Transactions on Graphics*.

Cranmer, M. (2023). Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint*.

Cranmer, M., Gonzalez, A. S., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. (2020a). Discovering symbolic models from deep learning with inductive biases. *Neural Information Processing Systems*.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. (2020b). Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*.

Cuomo, S., Cola, V. S. D., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing*.

da Costa-Ramos, L., Meglio, F. D., Morgenthaler, V., da Silva, L. F. F., and Bernard, P. (2020). Numerical design of luenberger observers for nonlinear systems. In *Conference on Decision and Control*.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. (2018). End-to-end differentiable physics for learning and control. *Neural Information Processing Systems*.

Devasia, S., Chen, D., and Paden, B. (1996). Nonlinear inversion-based output tracking. *Transactions on Automatic Control*.

Discetti, S. and Coletti, F. (2018). Volumetric velocimetry for fluid flows. *Measurement Science and Technology*.

Dissanayake, M. and Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representation*.

Draye, J.-P., Pavisic, D., Cheron, G., and Libert, G. (1996). Dynamic recurrent neural networks: a dynamical analysis. *Transactions on Systems, Man, and Cybernetics*.

Du, T., Wu, K., Ma, P., Wah, S., Spielberg, A., Rus, D., and Matusik, W. (2021). Diffpd: Differentiable projective dynamics. *Transactions on Graphics*.

Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes. *Neural Information Processing Systems*.

Dupont, E., Kim, H., Eslami, S., Rezende, D., and Rosenbaum, D. (2022a). From data to functa: Your data point is a function and you can treat it like one. *arXiv preprint*.

Dupont, E., Loya, H., Alizadeh, M., Golinski, A., Teh, Y. W., and Doucet, A. (2022b). Coin++: Data agnostic neural compression. *arXiv preprint*.

Dwivedi, V. and Srinivasan, B. (2020). Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations. *Neurocomputing*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

E, W. and Yu, B. (2017). The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*.

Eckert, M.-L., Um, K., and Thuerey, N. (2019). Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics*.

Ede, J. M. and Beanland, R. (2020). Adaptive learning rate clipping stabilizes learning. *Machine Learning: Science and Technology*.

Ehrhardt, S., Monszpart, A., Mitra, N., and Vedaldi, A. (2018). Unsupervised intuitive physics from visual observations. In *Asian Conference on Computer Vision*.

Ehsani, K., Tulsiani, S., Gupta, S., Farhadi, A., and Gupta, A. (2020). Use the force, luke! learning to predict physical forces by simulating effects. In *Conference on Computer Vision and Pattern Recognition*.

Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., and Kutz, J. N. (2020). Shallow neural networks for fluid flow reconstruction with limited sensors. *Royal Society A*.

Esmaeilzadeh, S., Azizzadenesheli, K., Kashinath, K., Mustafa, M., Tchelepi, H. A., Marcus, P., Prabhat, M., Anandkumar, A., et al. (2020). Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *International Conference for High Performance Computing, Networking, Storage and Analysis*.

Euler, L. (1824). *Institutionum calculi integralis*. Impensis Academia Imperialis Scientiarum.

Fang, Y. and Chow, T. W. (2005). Nonlinear dynamical systems control using a new rnn temporal learning strategy. *Transactions on Circuits and Systems*.

Fasel, U., Kutz, J. N., Brunton, B. W., and Brunton, S. L. (2022). Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Royal Society A*.

Fathony, R., Sahu, A. K., Willmott, D., and Kolter, J. Z. (2021). Multiplicative filter networks. In *International Conference on Learning Representations*.

Forgione, M., Mejari, M., and Piga, D. (2022). Learning neural state-space models: do we need a state estimator? *arXiv preprint*.

Francis, B. and Wonham, W. (1976). The internal model principle of control theory. *Automatica*.

Franz, E., Solenthaler, B., and Thuerey, N. (2021). Global transport for fluid reconstruction with learned self-supervision. In *Conference on Computer Vision and Pattern Recognition*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Funahashi, K. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*.

Gaby, N., Zhang, F., and Ye, X. (2022). Lyapunov-net: A deep neural network architecture for lyapunov function approximation. In *Conference on Decision and Control*.

Galerkin, B. G. (1915). Rods and plates. series occurring in various questions concerning the elastic equilibrium of rods and plates. *Engineers Bulletin*.

Galewsky, J., Scott, R. K., and Polvani, L. M. (2004). An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*.

Gao, H., Sun, L., and Wang, J.-X. (2021). Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*.

Gao, Z., Tan, C., Wu, L., and Li, S. Z. (2022). Simvp: Simpler yet better video prediction. In *Conference on Computer Vision and Pattern Recognition*.

Gedon, D., Wahlström, N., Schön, T. B., and Ljung, L. (2021). Deep state space models for nonlinear system identification. *International Federation of Automatic Control*.

Geilinger, M., Hahn, D., Zehnder, J., Bächer, M., Thomaszewski, B., and Coros, S. (2020). Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *Transactions on Graphics*.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *Transactions on Pattern Analysis and Machine Intelligence*.

Giaccagli, M., Astolfi, D., Andrieu, V., and Marconi, L. (2022a). Sufficient conditions for global integral action via incremental forwarding for input-affine nonlinear systems. *Transactions on Automatic Control*.

Giaccagli, M., Zoboli, S., Astolfi, D., Andrieu, V., and Casadei, G. (2022b). Synchronization in networks of nonlinear systems: Contraction metric analysis and deep-learning for feedback estimation. *Transaction on Automatic Control*.

Giftthaler, M., Neunert, M., Stäuble, M., Frigerio, M., Semini, C., and Buchli, J. (2017). Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*.

Gilpin, W. (2021). Chaos as an interpretable benchmark for forecasting and data-driven modelling. In *Neural Information Processing Systems*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Gojcic, Z., Litany, O., Wieser, A., Guibas, L. J., and Birdal, T. (2021). Weakly supervised learning of rigid 3d scene flow. In *Conference on Computer Vision and Pattern Recognition*.

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., et al. (2022). Kubric: A scalable dataset generator. In *Conference on Computer Vision and Pattern Recognition*.

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. *Neural Information Processing Systems*.

Guen, V. L. and Thome, N. (2020). Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Conference on Computer Vision and Pattern Recognition*.

Haber, E. and Ruthotto, L. (2017). Stable architectures for deep neural networks. *Inverse problems*.

Hairer, E., Nørsett, S. P., and Wanner, G. (1993). *Solving ordinary differential equations. 1, Nonstiff problems*. Springer-Vlg.

Han, X., Gao, H., Pfaff, T., Wang, J.-X., and Liu, L. (2021). Predicting physics in mesh-reduced space with temporal attention. In *International Conference on Learning Representations*.

Han, Y., Hao, W., and Vaidya, U. (2020). Deep learning of koopman representation for control. In *Conference on Decision and Control*.

Hao, Z., Ying, C., Su, H., Zhu, J., Song, J., and Cheng, Z. (2023). Bi-level physics-informed neural networks for pde constrained optimization using broyden's hypergradients. In *International Conference on Learning Representations*.

Hauser, J., Sastry, S., and Kokotovic, P. (1992). Nonlinear control via approximate input-output linearization: The ball and beam example. *Transaction on Automatic Control*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *International Conference on Computer Vision*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*.

Heinonen, M. and Lähdesmäki, H. (2019). Ode2vae: Deep generative second order odes with bayesian neural networks. *Neural Information Processing Systems*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

Holl, P., Thuerey, N., and Koltun, V. (2020). Learning to control pdes with differentiable physics. In *International Conference on Learning Representations*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*.

Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D., and Matusik, W. (2019). Chainqueen: A real-time differentiable physical simulator for soft robotics. In *International Conference on Robotics and Automation*.

Huang, B., Han, S. D., Boularias, A., and Yu, J. (2021). Dipn: Deep interaction prediction network with application to clutter removal. In *International Conference on Robotics and Automation*.

Isidori, A. (1995). *Nonlinear Control Systems*. Springer.

Isidori, A. and Byrnes, C. I. (1990). Output regulation of nonlinear systems. *Transactions on Automatic Control*.

Isidori, A., Praly, L., and Marconi, L. (2010). About the Existence of Locally Lipschitz Output Feedback Stabilizers for Nonlinear Systems. *Journal on Control and Optimization*.

Issa, R. I. (1986). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*.

Janny, S., Andrieu, V., Nadri, M., and Wolf, C. (2021). Deep kkl: Data-driven output prediction for non-linear systems. In *Conference on Decision and Control*.

Janny, S., Beneteau, A., Thome, N., Nadri, M., Digne, J., and Wolf, C. (2023). Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *International Conference on Learning Representation*.

Janny, S., Possamaï, Q., Bako, L., Wolf, C., and Nadri, M. (2022). Learning reduced nonlinear state-space models: an output-error based canonical approach. In *Conference on Decision and Control*.

Janot, A., Gautier, M., and Brunot, M. (2019). Data set and reference models of emps. In *Nonlinear System Identification Benchmarks*.

Jaques, M., Burke, M., and Hospedales, T. (2020). Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *International Conference on Learning Representations*.

Jia, J. and Benson, A. R. (2019). Neural jump stochastic differential equations. *Neural Information Processing Systems*.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Karlbauer, M., Praditia, T., Otte, S., Oladyshkin, S., Nowak, W., and Butz, M. V. (2022). Composing partial differential equations with physics-aware neural networks. In *International Conference on Machine Learning*.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*.

Kaufmann, E., Bauersfeld, L., Loquercia, A., Müller, M., Koltun, V., and Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*.

Kazantzis, N. and Kravaris, C. (1998). Nonlinear observer design using lyapunov's auxiliary theorem. *Systems & Control Letters*.

Kervadec, C., Jaunet, T., Antipov, G., Baccouche, M., Vuillemot, R., and Wolf, C. (2021). How transferable are reasoning patterns in vqa? In *Conference on Computer Vision and Pattern Recognition*.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. (2019). Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2018). Attentive neural processes. In *International Conference on Learning Representations*.

Kim, S.-W. and Benson, T. (1992). Comparison of the smac, piso and iterative time-advancing schemes for unsteady flows. *Computers & fluids*.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kirchmeyer, M., Yin, Y., Donà, J., Baskiotis, N., Rakotomamonjy, A., and Gallinari, P. (2022). Generalizing to new physical systems via context-informed dynamics model. In *International Conference on Machine Learning*.

Klushyn, A., Kurle, R., Soelch, M., Cseke, B., and van der Smagt, P. (2021). Latent matters: Learning deep state-space models. *Neural Information Processing Systems*.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *National Academy of Sciences*.

Kohl, G., Um, K., and Thuerey, N. (2020). Learning similarity metrics for numerical simulations. In *International Conference on Machine Learning*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Kolter, J. Z. and Manek, G. (2019). Learning stable deep dynamics models. *Neural Information Processing Systems*.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Neural operator: Learning maps between function spaces. *arXiv preprint*.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Neural Information Processing Systems*.

Krueger, D. and Memisevic, R. (2016). Regularizing rnns by stabilizing activations. In *International Conference on Learning Representation*.

Kulkarni, T. D., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A., and Mnih, V. (2019). Unsupervised learning of object keypoints for perception and control. *Neural Information Processing Systems*.

Kurz, M., Offenhäuser, P., and Beck, A. (2023). Deep reinforcement learning for turbulence modeling in large eddy simulations. *International Journal of Heat and Fluid Flow*.

Kutta, W. (1901). *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. Teubner.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *Transactions on Neural Networks*.

Lagaris, I. E., Likas, A. C., and Papageorgiou, D. G. (2000). Neural-network methods for boundary value problems with irregular boundaries. *Transactions on Neural Networks*.

Lee, A. X., Gupta, A., Lu, H., Levine, S., and Abbeel, P. (2015). Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation. In *International Conference on Intelligent Robots and Systems*.

Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., and Levine, S. (2018). Stochastic adversarial video prediction. In *International Conference on Learning Representation*.

Lerer, A., Gross, S., and Fergus, R. (2016). Learning physical intuition of block towers by example. In *International Conference on Machine Learning*.

Li, M. and Jiang, L. (2021). Deep learning nonlinear multiscale dynamic problems using koopman operator. *Journal of Computational Physics*.

Li, S., Huang, Z., Chen, T., Du, T., Su, H., Tenenbaum, J. B., and Gan, C. (2023). Dexdeform:

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Dexterous deformable object manipulation with human demonstrations and differentiable physics. In *International Conference on Learning Representations*.

Li, Y., Perlman, E., Wan, M., Yang, Y., Meneveau, C., Burns, R., Chen, S., Szalay, A., and Eyink, G. (2008). A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*.

Li, Y., Torralba, A., Anandkumar, A., Fox, D., and Garg, A. (2020a). Causal discovery in physical systems from videos. In *Neural Information Processing Systems*.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. (2018). Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2020b). Neural operator: Graph kernel network for partial differential equations. *arXiv preprint*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Markov neural operators for learning chaotic systems. *arXiv preprint*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. (2020c). Multipole graph neural operator for parametric partial differential equations. In *Neural Information Processing Systems*.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A., et al. (2020d). Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*.

Liao, Y. and Ming, P. (2019). Deep nitsche method: Deep ritz method with essential boundary conditions. *arXiv preprint*.

Liberzon, D. (2011). *Calculus of Variations and Optimal Control Theory*. Daniel Liberzon.

Lienen, M. and Günnemann, S. (2022). Learning the dynamics of physical systems from sparse observations with finite element networks. In *International Conference on Learning Representations*.

Lino, M., Cantwell, C., Bharath, A. A., and Fotiadis, S. (2021). Simulating continuum mechanics with multi-scale graph neural networks. *arXiv preprint*.

Liu, Z., Chen, Y., Du, Y., and Tegmark, M. (2021). Physics-augmented learning: A new paradigm beyond physics-informed learning. *arXiv preprint*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Ljung, L., Andersson, C., Tiels, K., and Schön, T. B. (2020). Deep learning and system identification. *International Federation of Automatic Control*.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. In *Neural Information Processing Systems*.

Lohmiller, W. and Slotine, J. E. (1998). On contraction analysis for non-linear systems. *Automatica*.

Long, Y., She, X., and Mukhopadhyay, S. (2018). Hybridnet: Integrating model-based and data-driven learning to predict evolution of dynamical systems. In *Conference on Robot Learning*.

Long, Z., Lu, Y., and Dong, B. (2019). Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*.

Loquercio, A., Kaufmann, E., Ranftl, R., Müller, M., Koltun, V., and Scaramuzza, D. (2021). Learning high-speed flight in the wild. *Science Robotics*.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *International Conference on Computer Vision*.

Lu, L., Jin, P., and Karniadakis, G. E. (2019). Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint*.

Lu, Y., Chen, Y., Zhao, D., and Li, D. (2021). Mgrl: Graph neural network based inference in a markov network with reinforcement learning for visual navigation. *Neurocomputing*.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The Expressive Power of Neural Networks: A View from the Width. In *Neural Information Processing Systems*.

Luenberger, D. (1971). An introduction to observers. *Transactions on Automatic Control*.

Luenberger, D. G. (1964). Observing the state of a linear system. *Transactions on Military Electronics*.

Lusch, B., Kutz, J. N., and Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*.

Lutter, M., Ritter, C., and Peters, J. (2019). Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Ma, P., Du, T., Zhang, J. Z., Wu, K., Spielberg, A., Katzschmann, R. K., and Matusik, W. (2021). Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *Transactions on Graphics*.

Mahony, R., Kumar, V., and Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *Robotics & Automation Magazine*.

Mai, V. V. and Johansson, M. (2021). Stability and convergence of stochastic gradient clipping: Beyond lipschitz continuity and smoothness. In *International Conference on Machine Learning*.

Manchester, I. and Slotine, J. (2017). Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *Transaction on Automatic Control*.

Manuelli, L., Gao, W., Florence, P., and Tedrake, R. (2019). kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*.

Manuelli, L., Li, Y., Florence, P., and Tedrake, R. (2021). Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning*.

Marconi, L., Praly, L., and Isidori, A. (2007). Output Stabilization via Nonlinear Luenberger Observers. *Journal on Control and Optimization*.

Mardt, A., Pasquali, L., Noé, F., and Wu, H. (2020). Deep learning markov and koopman models with physical constraints. In *Mathematical and Scientific Machine Learning*.

Margenberg, N., Hartmann, D., Lessig, C., and Richter, T. (2022). A neural network multigrid solver for the navier-stokes equations. *Journal of Computational Physics*.

Martin-Ordas, G., Call, J., and Colmenares, F. (2008). Tubes, tables and traps: great apes solve twofunctionally equivalent trap tasks but show no evidence of transfer across tasks. *Animal Cognition*.

Masti, D. and Bemporad, A. (2018). Learning nonlinear state-space models using deep autoencoders. In *Conference on Decision and Control*.

Matsunaga, D., Suzumura, T., and Takahashi, T. (2019). Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint*.

McClenny, L. and Braga-Neto, U. (2020). Self-adaptive physics-informed neural networks using a soft attention mechanism. In *arXiv preprint*.

McFall, K. S. and Mahan, J. R. (2009). Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *Transactions on Neural Networks*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Meer, R. V. D., Oosterlee, C., and Borovykh, A. (2022). Optimally weighted loss functions for solving pdes with neural networks. *Journal of Computational and Applied Mathematics*.

Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., and Chandraker, M. (2021a). Modulated periodic activations for generalizable local functional representations. In *International Conference on Computer Vision*.

Mehta, V., Char, I., Neiswanger, W., Chung, Y., Nelson, A., Boyer, M., Kolemen, E., and Schneider, J. (2021b). Neural dynamical systems: Balancing structure and flexibility in physical prediction. In *Conference on Decision and Control*.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Conference on Computer Vision and Pattern Recognition*.

Mezić, I. (2005). Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*.

Mezić, I. and Banaszuk, A. (2004). Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*.

Miao, K. and Gatsis, K. (2023). Learning robust state observers using neural odes. In *Learning for Dynamics and Control Conference*.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*.

Minderer, M., Sun, C., Villegas, R., Cole, F., Murphy, K. P., and Lee, H. (2019). Unsupervised learning of object structure and dynamics from videos. In *Neural Information Processing Systems*.

Misyris, G. S., Venzke, A., and Chatzivasileiadis, S. (2020). Physics-informed neural networks for power systems. In *Power & Energy Society General Meeting*.

Mohan, A. T., Lubbers, N., Livescu, D., and Chertkov, M. (2020). Embedding hard physical constraints in neural network coarse-graining of 3d turbulence. *arXiv preprint*.

Nagabandi, A., Konolige, K., Levine, S., and Kumar, V. (2020). Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*.

Noack, B. R., Afanisiev, K., Morzynski, M., Tadmor, G., and Thiele, F. (2003). A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. (2020). On second order behaviour in augmented neural odes. *Neural Information Processing Systems*.

Nutkiewicz, A., Yang, Z., and Jain, R. K. (2018). Data-driven urban energy simulation (due-s): A framework for integrating engineering simulation and machine learning methods in a multi-scale urban energy modeling workflow. *Applied energy*.

Obiols-Sales, O., Vishnu, A., Malaya, N., and Chandramowliswharan, A. (2020). Cfdnet: A deep learning-based accelerator for fluid simulations. In *International Conference on Supercomputing*.

Pan, Y. and Wang, J. (2011). Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *Transactions on Industrial Electronics*.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Conference on Computer Vision and Pattern Recognition*.

Pavlov, A., Wouw, N. V. D., and Nijmeijer, H. (2006). *Uniform output regulation of nonlinear systems: a convergent dynamics approach*. Springer.

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Pearl, J. (2018). Causal and counterfactual inference. *The Handbook of Rationality*.

Peralez, J., Galuppo, F., Dufour, P., Wolf, C., and Nadri, M. (2020). Data-driven multimodel control waste for heat recovery system on a heavy duty truck engine. In *Conference on Decision and Control*.

Peralez, J. and Nadri, M. (2021). Deep learning-based luenberger observer design for discrete-time nonlinear systems. In *Conference on Decision and Control*.

Peralez, J., Nadri, M., and Astolfi, D. (2022). Neural network-based kkl observer for nonlinear discrete-time systems. In *Conference on Decision and Control*.

Petridis, V. and Petridis, S. (2006). Construction of neural network based lyapunov functions. In *International Joint Conference on Neural Network Proceedings*.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. (2020). Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*.

Pfaff, T. and Thuerey, N. (2016). Mantaflow.

Pfeiffer, C., Wengeler, S., Loquercio, A., and Scaramuzza, D. (2022). Visual attention prediction improves performance of autonomous drone racing agents. *Plos one*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G., and Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. (2019). Graph neural ordinary differential equations. *arXiv preprint*.

Poli, M., Massaroli, S., Scimeca, L., Chun, S., Oh, S. J., Yamashita, A., Asama, H., Park, J., and Garg, A. (2021). Neural hybrid automata: Learning dynamics with multiple modes and stochastic transitions. *Neural Information Processing Systems*.

Pontryagin, L. S. (1987). *Mathematical theory of optimal processes*. CRC press.

Possamaï, Q., Janny, S., Nadri, M., Bako, L., and Wolf, C. (2022). Learning to estimate uav created turbulence from scene structure observed by onboard cameras. In *ArXiv pre-print*.

Prouty, R. W. (1995). *Helicopter performance, stability, and control*. R.E. Krieger Publishing Company.

Psichogios, D. C. and Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling. *AIChE Journal*.

Qiao, Y.-L., Liang, J., Koltun, V., and Lin, M. (2020). Scalable differentiable physics for learning and control. In *International Conference on Machine Learning*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint*.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. *Neural Information Processing Systems*.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint*.

Ren, J., Yu, C., Chen, S., Ma, X., Pan, L., and Liu, Z. (2023). Diffmimic: Efficient motion mimicking with differentiable physics. In *International Conference on Learning Representations*.

Revach, G., Shlezinger, N., Ni, X., Escoriza, A. L., Sloun, R. J. V., and Eldar, Y. C. (2022). Kalmannet: Neural network aided kalman filtering for partially known dynamics. *Transactions on Signal Processing*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Rico-Martinez, R., Anderson, J., and Kevrekidis, I. (1994). Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In *Workshop on Neural Networks for Signal Processing*.

Rodriguez, I. D. J., Ames, A., and Yue, Y. (2022). Lyanet: A lyapunov framework for training neural odes. In *International Conference on Machine Learning*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition*.

Romero, A., Sun, S., Foehn, P., and Scaramuzza, D. (2022). Model predictive contouring control for time-optimal quadrotor flight. *Transactions on Robotics*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*.

Rowley, C. W., Mezic, I., Bagheri, S., Schlatter, P., and Henningson, D. S. (2009). Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*.

Ruchti, T. L., Brown, R. H., and Garside, J. J. (1993). Kalman based artificial neural network training algorithms for nonlinear system identification. In *International Symposium on Intelligent Control*.

Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2017). Data-driven discovery of partial differential equations. *Science Advances*.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1985). Learning internal representations by error propagation.

Runge, C. (1895). Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*.

Ryoo, M. S., Gopalakrishnan, K., Kahatapitiya, K., Xiao, T., Rao, K., Stone, A., Lu, Y., Ibarz, J., and Arnab, A. (2023). Token turing machines. In *Conference on Computer Vision and Pattern Recognition*.

Ryu, H., in Lee, H., Lee, J.-H., and Choi, J. (2022). Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. In *International Conference on Learning Representations*.

Saad, N., Gupta, G., Alizadeh, S., and Maddix, D. C. (2022). Guiding continuous operator learning through physics-based boundary constraints. In *International Conference on Learning Representations*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Saha, A., Mendez, O., Russell, C., and Bowden, R. (2022). Translating images into maps. In *International Conference on Robotics and Automation*.

Sahoo, S., Lampert, C., and Martius, G. (2018). Learning equations for extrapolation and control. In *International Conference on Machine Learning*.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. In *International Conference on Machine Learning*.

Scaman, K. and Virmaux, A. (2018). Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Neural Information Processing Systems*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *Transactions on Neural Networks*.

Schaeffer, H. (2017). Learning partial differential equations via data discovery and sparse optimization. *Royal Society A: Mathematical, Physical and Engineering Sciences*.

Schmidt, D., Koppe, G., Monfared, Z., Beutelspacher, M., and Durstewitz, D. (2020). Identifying nonlinear dynamical systems with multiple time scales and long-range dependencies. In *International Conference on Learning Representations*.

Schmidt, M. and Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*.

Schoukens, M. and Noël, J. P. (2017). Three benchmarks addressing open challenges in nonlinear system identification. *International Federation of Automatic Control*.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*.

Searson, D. P., Leahy, D. E., and Willis, M. J. (2010). Gptips: an open source genetic programming toolbox for multigene symbolic regression. In *International multiconference of engineers and computer scientists*.

Seita, D., Ganapathi, A., Hoque, R., Hwang, M., Cen, E., Tanwani, A. K., Balakrishna, A., Thananjeyan, B., Ichnowski, J., Jamali, N., et al. (2020). Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor. In *International Conference on Intelligent Robots and Systems*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Sekar, V., Jiang, Q., Shu, C., and Khoo, B. C. (2019). Fast flow field prediction over airfoils using deep learning approach. *Physics of Fluids*.

Serrani, A., Isidori, A., and Marconi, L. (2001). Semi-global nonlinear output regulation with adaptive internal model. *Transactions on Automatic Control*.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press.

Sheng, H. and Yang, C. (2021). Pfnn: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. *Journal of Computational Physics*.

Shi, G., Shi, X., O'Connell, M., Yu, R., Azizzadenesheli, K., Anandkumar, A., Yue, Y., and Chung, S.-J. (2019). Neural lander: Stable drone landing control using learned dynamics. In *International Conference on Robotics and Automation*.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and chun Woo, W. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Neural Information Processing Systems*.

Shlomi, J., Battaglia, P., and Vlimant, J.-R. (2020). Graph neural networks in particle physics. *Machine Learning: Science and Technology*.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Neural Information Processing Systems*.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing*.

Song, Y., Shi, K., Penicka, R., and Scaramuzza, D. (2023). Learning perception-aware agile flight in cluttered environments. In *International Conference on Robotics and Automation*.

Sontag, E. D. (2010). Contractive systems with inputs. *Perspectives in Mathematical System Theory, Control, and Signal Processing*.

Sontag, E. D. and Wang, Y. (1995). On characterizations of the input-to-state stability property. *Systems & Control Letters*.

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. (2021). Learned simulators for turbulence. In *International Conference on Learning Representations*.

Stachenfeld, K., Godwin, J., and Battaglia, P. (2020). Graph networks with spectral message passing. *arXiv preprint*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Stam, J. (1999). Stable fluids. In *Seminal Graphics Papers: Pushing the Boundaries*.

Stokes, G. G. (2009). *On the Effect of the Internal Friction of Fluids on the Motion of Pendulums*. Cambridge University Press.

Sun, F., Liu, Y., Wang, J.-X., and Sun, H. (2023). Symbolic physics learner: Discovering governing equations via monte carlo tree search. In *International Conference on Learning Representations*.

Takahashi, T., Liang, J., Qiao, Y.-L., and Lin, M. C. (2021). Differentiable fluids with solid coupling for learning and control. In *Conference on Artificial Intelligence*.

Takeishi, N. and Kalousis, A. (2021). Physics-integrated variational autoencoders for robust and interpretable generative modeling. *Neural Information Processing Systems*.

Thompson, M. L. and Kramer, M. A. (1994). Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*.

Thuerey, N., Weißenow, K., Prantl, L., and Hu, X. (2020). Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *American Institute of Aeronautics and Astronautics*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*.

Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. (2017). Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*.

Toth, P., Rezende, D. J., Jaegle, A., Racanière, S., Botev, A., and Higgins, I. (2019). Hamiltonian generative networks. *arXiv preprint*.

Toussaint, M., Allen, K. R., Smith, K. A., and Tenenbaum, J. B. (2019). Differentiable physics and stable modes for tool-use and manipulation planning–extended abstract. In *Joint Conference on Artificial Intelligence*.

Tran, G. Q. B. and Bernard, P. (2023). Arbitrarily fast robust kkl observer for nonlinear time-varying discrete systems. *HAL*.

Um, K., Brand, R., Fei, Y. R., Holl, P., and Thuerey, N. (2020). Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Neural Information Processing Systems*.

Valiant, L. (1984). A theory of the learnable. In *Communications of the ACM*.

van der Pol Jun. D.Sc, B. (1926). On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Neural Information Processing Systems*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. In *International Conference on Learning Representation*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Versteeg, H. K. and Malalasekera, W. (2007). *An introduction to computational fluid dynamics: the finite volume method*. Pearson education.

Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., and Fragkiadaki, K. (2017). Sfm-net: Learning of structure and motion from video. *arXiv preprint*.

Volterra, V. and Brelot, M. (1931). *Leçons sur la théorie mathématique de la lutte pour la vie*. Gauthier-Villars et cie.

Wandel, N., Weinmann, M., and Klein, R. (2021). Learning Incompressible Fluid Dynamics from Scratch – Towards Fast, Differentiable Fluid Models that Generalize. In *International Conference on Learning Representations*.

Wang, J.-S. and Chen, Y.-P. (2006). A fully automated recurrent neural network for unknown dynamic system identification and control. *Transactions on Circuits and Systems*.

Wang, Q., Li, F., Tang, Y., and Xu, Y. (2019a). Integrating model-driven and data-driven methods for power system frequency stability assessment and control. *Transactions on Power Systems*.

Wang, Q., Li, F., Tang, Y., and Xu, Y. (2019b). Integrating model-driven and data-driven methods for power system frequency stability assessment and control. *Transactions on Power Systems*.

Wang, S., Sankaran, S., and Perdikaris, P. (2022a). Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint*.

Wang, S., Teng, Y., and Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *Journal on Scientific Computing*.

Wang, S., Yu, X., and Perdikaris, P. (2022b). When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*.

Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *International Conference on Learning Representation*.

Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.-S. (2019c). Kgat: Knowledge graph attention

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

network for recommendation. In *International Conference on Knowledge Discovery & Data Mining*.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. (2019d). Heterogeneous graph attention network. In *The World Wide Web Conference*.

Wang, Y., Idoughi, R., and Heidrich, W. (2020). Stereo event-based particle tracking velocimetry for 3d fluid flow reconstruction. In *European Conference on Computer Vision*.

Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P. S. (2017). Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *Neural Information Processing Systems*.

Wang, Y., Wang, J., Cao, Z., and Farimani, A. B. (2022c). Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*.

Wang, Y., Wu, H., Zhang, J., Gao, Z., Wang, J., Philip, S. Y., and Long, M. (2022d). Predrnn: A recurrent neural network for spatiotemporal predictive learning. *Transactions on Pattern Analysis and Machine Intelligence*.

Wang, Y., Zhang, J., Zhu, H., Long, M., Wang, J., and Yu, P. S. (2019e). Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. In *Conference on Computer Vision and Pattern Recognition*.

Wang, Y.-J. and Lin, C.-T. (1998). Runge-kutta neural network for identification of dynamical systems in high accuracy. *Transactions on Neural Networks*.

Weigand, J., Götz, J., Ulmen, J., and Ruskowski, M. (2023). Dataset and baseline for an industrial robot identification benchmark. In *Workshop on Nonlinear System Identification Benchmarks*.

Weinmann, A. (2012). *Uncertain models and robust control*. Springer Science & Business Media.

Wiewel, S., Becher, M., and Thuerey, N. (2019). Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*.

Wight, C. L. and Zhao, J. (2020). Solving allen–cahn and cahn–hilliard equations using the adaptive physics informed neural networks. *arXiv preprint*.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*.

Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. (2017a). Learning to see physics via visual de-animation. In *Neural Information Processing Systems*.

Wu, X., Moin, P., Wallace, J. M., Skarda, J., Lozano-Durán, A., and Hickey, J.-P. (2017b). Tran-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

sitional turbulent spots and turbulent turbulent spots in boundary layers. *National Academy of Sciences*.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. (2020). On layer normalization in the transformer architecture. In *International Conference on Machine Learning*.

Xu, Z., Wu, J., Zeng, A., Tenenbaum, J. B., and Song, S. (2019). Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint*.

Yeung, E., Kundu, S., and Hodas, N. (2019). Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *American Control Conference*.

Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. (2019). Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*.

Yin, Y., Ayed, I., de Bézenac, E., Baskiotis, N., and Gallinari, P. (2021a). Leads: Learning dynamical systems that generalize across environments. *Neural Information Processing Systems*.

Yin, Y., Guen, V. L., Dona, J., de Bézenac, E., Ayed, I., Thome, N., and Gallinari, P. (2021b). Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*.

Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., et al. (2022). Continuous pde dynamics forecasting with implicit neural representations. In *International Conference on Learning Representations*.

Young, C.-C., Liu, W.-C., and Wu, M.-C. (2017). A physically based and machine learning hybrid approach for accurate rainfall-runoff modeling during extreme typhoon events. *Applied Soft Computing*.

Yu, C., Bi, X., and Fan, Y. (2023). Deep learning for fluid velocity field estimation: A review. *Ocean Engineering*.

Zang, Y., Bao, G., Ye, X., and Zhou, H. (2020). Weak adversarial networks for highdimensional partial differential equations. *Journal of Computational Physics*.

Zeng, A., Florence, P., Tompson, J., Welker, S., Chien, J., Attarian, M., Armstrong, T., Krasin, I., Duong, D., Sindhwani, V., et al. (2021). Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*.

Zeng, Q., Kothari, Y., Bryngelson, S. H., and Schäfer, F. (2023). Competitive physics informed networks. In *International Conference on Learning Representations*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Zhang, J., Henein, M., Mahony, R., and Ila, V. (2020a). Vdo-slam: a visual dynamic object-aware slam system. *arXiv preprint*.

Zhang, W., Liu, H., Liu, Y., Zhou, J., Xu, T., and Xiong, H. (2020b). Semi-supervised city-wide parking availability prediction via hierarchical recurrent graph neural network. *Transactions on Knowledge and Data Engineering*.

Zhang, Z., Zhuang, F., Zhu, H., Shi, Z., Xiong, H., and He, Q. (2020c). Relational graph neural network with hierarchical attention for knowledge graph completion. In *Conference on Artificial Intelligence*.

Zhao, T., Zheng, Y., Gong, J., and Wu, Z. (2022). Machine learning-based reduced-order modeling and predictive control of nonlinear processes. *Chemical Engineering Research and Design*.

Zhong, Y. D., Dey, B., and Chakraborty, A. (2019). Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*.

Zhong, Y. D. and Leonard, N. (2020). Unsupervised learning of lagrangian dynamics from images for prediction and control. In *Neural Information Processing Systems*.

Zhu, M., Derpanis, K. G., Yang, Y., Brahmbhatt, S., Zhang, M., Phillips, C., Lecce, M., and Daniilidis, K. (2014). Single image 3d object detection and pose estimation for grasping. In *International Conference on Robotics and Automation*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Part V

Appendices

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Appendix A

Appendix of chapter 4

A.1 Proof of proposition 4.2

It follows from the definition (4-10) of $\hat{\boldsymbol{s}}[k-\ell]$ that

$$\left|\bar{\boldsymbol{y}}[k-\ell|k-1] - \mathcal{O}_\ell\big(\hat{\boldsymbol{s}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\right| \leqslant \left|\bar{\boldsymbol{y}}[k-\ell|k-1] - \mathcal{O}_\ell\big(\boldsymbol{s}, \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\right|, \tag{A-1}$$

for all $\boldsymbol{s} \in \mathcal{S}$. In particular, this inequality holds for $\boldsymbol{s} = \boldsymbol{s}[k-\ell]$. By then invoking equation (4-9) we get

$$\left|\mathcal{O}_\ell\big(\boldsymbol{s}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) - \mathcal{O}_\ell\big(\hat{\boldsymbol{s}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) + \bar{\boldsymbol{w}}[k-\ell|k-1]\right| \leqslant \left|\bar{\boldsymbol{w}}[k-\ell|k-1]\right|. \tag{A-2}$$

By the triangle inequality, it follows that

$$\begin{aligned} &\left|\mathcal{O}_\ell\big(\boldsymbol{s}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) - \mathcal{O}_\ell\big(\hat{\boldsymbol{s}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) + \bar{\boldsymbol{w}}[k-\ell|k-1]\right| \\ \geqslant &\left|\mathcal{O}_\ell\big(\boldsymbol{s}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) - \mathcal{O}_\ell\big(\hat{\boldsymbol{x}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\right| - \left|\bar{\boldsymbol{w}}[k-\ell|k-1]\right|. \end{aligned} \tag{A-3}$$

Using the uniform observability,

$$\begin{aligned} \alpha_\ell\left|\hat{\boldsymbol{s}}[k-\ell] - \boldsymbol{s}[k-\ell]\right| &\leqslant \left|\mathcal{O}_\ell\big(\boldsymbol{s}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big) - \mathcal{O}_\ell\big(\hat{\boldsymbol{s}}[k-\ell], \bar{\boldsymbol{u}}[k-\ell|k-1]\big)\right| \\ &\leqslant 2\left|\bar{\boldsymbol{w}}[k-\ell|k-1]\right|. \end{aligned} \tag{A-4}$$

As a consequence, we get $\left|\hat{\boldsymbol{s}}[k-\ell] - \boldsymbol{s}[k-\ell]\right| \leqslant 2\alpha_\ell^{-1}\left|\bar{\boldsymbol{w}}[k-\ell|k-1]\right|$.

A.2 Model details

Classic GRU – is a 2-layer Gated Recurrent Unit (GRU). The hidden vector size is chosen such that the cumulated dimension of the two hidden vectors matches the one of the corresponding state $\boldsymbol{z}[k]$, formally $n_h = \frac{l}{2}(n_u + n_y)$. The hidden vector is then decoded by an MLP with one hidden unit of size n_h.

Ours (MLP) – uses an MLP to model H with 3 hidden units of size 256 for the simulated datasets and 2-layer with 2048 units for the 3D Drone. The encoder-decoder is modeled with two MLP of 2 layers of 512 units.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Ours (GRU) – model H with a GRU with three layers, and hidden size of 128. The encoder-decoder is identical to *Ours (MLP)*.

Masti and Bemporad (2018) – uses a 3-layered MLP with 256 neurons per hidden layer for the dynamics, and 2-layered, 128 neurons MLPs for the encoder and the decoder.

Each model is implemented in PyTorch and trained with Adam optimizer, with a 10^{-4} learning rate. We trained the regressors for 10,000 epochs, and the encoder for 3,000 epochs on the simulated datasets and respectively 300 epochs on the 3D drone dataset.

A.3 Dataset details

Tank dataset is composed of trajectories generated by uniformly sampling five waypoints in $[0, 5]$ evenly distributed on time to construct a 200 steps reference signal, using cubic spline interpolation. This reference is then tracked with a PID controller. The dataset contains 60 trajectories for the train set and 20 for both the validation and test set. We use $k_1 = 0.5, k_2 = 0.4, k_3 = 0.2$ and $k_1 = 0.3$ for simulation.

3D drone dataset built on the BlackBird dataset (Antonini et al., 2018). We extracted IMU measurements and commands from raw flight data and applied pre-processing as follows: temporal synchronization of both signals, noise filtering using Butterworth filters, and sampling rate reduction to 50Hz. To create train/valid/test splits, we sampled 20 flights to create the validation split and 10 for the test split. The remaining 146 flights were used for the training set. Each flight has been sliced into 200-step chunks to ease training.

2D drone dataset is composed of 2D flights generated by uniformly sampling five to ten 2D waypoints in $[-2, 2]^2$ evenly distributed on time to construct a 600-step reference signal using cubic spline interpolation. This reference is then tracked with Model Predictive Control. The dataset contains 500 flights for the train set and 20 flights for validation and test sets. For simulation, we choose $k_T = 4 \times 10^{-4}, \gamma = 10^{-9}, L = 0.15, m = 1$ and $J = 2.7 \times 10^{-3}$. The system is simulated with an Euler integration scheme at 30Hz.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Appendix B

Appendix of chapter 5

B.1 Proof of Proposition 5.1

Note that

$$Z(z_0^y, \ell, y) = \mathcal{Z}(z_0^y, \ell). \tag{B-1}$$

Hence, with the contraction property (5.1), it gives :

$$|Z(0, \ell, y) - \mathcal{Z}(z_0^y, \ell)| \leqslant k e^{-\lambda \ell} |z_0^y| \,. \tag{B-2}$$

Due to the Lipschitz property, it yields for all (z_a, z_b) and all $p \geqslant 0$

$$|\mathcal{Z}(z_a, p) - \mathcal{Z}(z_b, p)| \leqslant e^{L_1 p} |z_a - z_b| \,. \tag{B-3}$$

Setting $z_a = Z(0, \ell, y)$ and $z_b = \mathcal{Z}(z_0^y, \ell)$, the former inequality becomes

$$\begin{aligned} |\mathcal{Z}(Z(0, \ell, y), p) - \mathcal{Z}(z_0^y, \ell + p)| &\leqslant e^{L_1 p} |Z(0, \ell, y) - \mathcal{Z}(z_0^y, \ell)| \,, \\ &\leqslant k e^{-\lambda \ell + L_1 p} |z_0^y| \end{aligned} \tag{B-4}$$

Since (g, ψ) is a generating model, and since (5-21) holds, it yields

$$\begin{aligned} |\hat{y}(\ell + p) - y(\ell + p)| = |\psi(\mathcal{Z}(Z(0, \ell, y), p)) - \psi(\mathcal{Z}(z_0^y, \ell + p))|, \\ \leqslant L_2 k e^{-\lambda \ell + L_1 p} |z_0^y| \,. \end{aligned} \tag{B-5}$$

B.2 Proof of Theorem 5.1

Theorem 5.1 mostly relies on the results obtained in Andrieu and Praly (2006) in the context of observer design and Marconi et al. (2007) in the context of output regulation. The proof of this statement relies on the existence of a C^1 function $T : \mathcal{O} \mapsto \mathbb{R}^m$ mapping s to z which satisfies the differential equation :

$$L_f T(s) = A T(s) + b h(s), \quad \forall s \in \mathcal{O} \,, \tag{B-6}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

where $L_f T$ is the Lie derivative of T along f. The functions ψ and T need to satisfy the equality

$$\psi(T(\boldsymbol{s})) = h(\boldsymbol{s}), \quad \forall \boldsymbol{s} \in \mathcal{O} \,. \tag{B-7}$$

Given a Hurwitz matrix A, as shown in Andrieu and Praly (2006), the following function T

$$T(\boldsymbol{s}) = \int_{-\infty}^{0} e^{-At} \boldsymbol{b} h(S(\boldsymbol{s}, t)) dt \,, \tag{B-8}$$

is well defined for $\boldsymbol{s}$ in $\mathcal{O}$ and satisfies equation (B-6). It can be shown that T is C^1 if the eigenvalues of A are smaller than a specific value depending on the Lipschitz constant of f. The proof of these results is detailed in Andrieu and Praly (2006) (see Theorem 2.4). To find a function ψ such that equation (B-7) is satisfied, we need to ensure that T contains enough information to represent the observation y. This requirement can be expressed as a pseudo-injectivity with regards to h :

$$\forall (\boldsymbol{s}_1, \boldsymbol{s}_2) \in \mathcal{O} \quad T(\boldsymbol{s}_1) = T(\boldsymbol{s}_2) \Rightarrow h(\boldsymbol{s}_1) = h(\boldsymbol{s}_2). \tag{B-9}$$

It is shown in (Marconi et al., 2007, Proposition 2) that this condition is satisfied provided $m = 2(n+1)$ and A is the real representation of a Hurwitz diagonal matrix. Finally, (Marconi et al., 2007, Proposition 3) states the existence of ψ. In conclusion, if the dimension of $\boldsymbol{z} \in \mathbb{R}^m$ is greater or equal to $m = 2n + 2$, then there exists a continuous function $\psi : \mathbb{R}^m \mapsto \mathbb{R}$ such that for any trajectory y in $\mathcal{Y}_{\mathcal{O}}$, there exists z_0^y such that:

$$\begin{aligned} &\dot{\boldsymbol{z}} = A\boldsymbol{z} + \boldsymbol{b}y \quad \boldsymbol{z}(0) = z_0^y, \\ &\psi\big(Z(z_0^y, t, y)\big) = y(t), \quad \forall t \,. \end{aligned} \tag{B-10}$$

B.3 Proof of Proposition 5.2

The proof of Proposition 5.2 relies mostly on the results presented in Andrieu (2014). We follow the steps of the proof of Theorem 5.1. However, it is shown in (Andrieu, 2014, Proposition 3.5) and (Andrieu, 2014, Proposition 3.6) that if $m = 2n + 2$, there exist $(A, \boldsymbol{b})$ such that the function T given in (B-8) is injective and full rank in $\mathcal{O}$. Employing (Andrieu, 2014, Lemma 3.2), we obtain the existence of a positive real number L_T such that

$$L_T |T(\boldsymbol{s}_1) - T(\boldsymbol{s}_2)| \geqslant |\boldsymbol{s}_1 - \boldsymbol{s}_2|, \quad \forall (\boldsymbol{s}_1, \boldsymbol{s}_2) \in \mathcal{O} \,. \tag{B-11}$$

Hence, denoting L_3 the Lipschitz constant of h, for all (z_1, z_2) in $T(\mathcal{O})^2$, it yields

$$\begin{aligned} |h(T^{-1}(z_1)) - h(T^{-1}(z_2))| &\leqslant L_3 |T^{-1}(z_1) - T^{-1}(z_2)| \,, \\ &\leqslant L_3 L_T |z_1 - z_2| \,. \end{aligned} \tag{B-12}$$

Defining ψ as a global Lipschitz extension of $h \circ T^{-1}$ to $\mathbb{R}^m$ yields the first and second part with $L_2 = L_3 L_T$ of the proposition. The third part of the Proposition is simply obtained by noticing that with $g(\boldsymbol{z}) = A\boldsymbol{z} + \boldsymbol{b}\psi(\boldsymbol{z})$,

$$\left| \frac{\partial g}{\partial \boldsymbol{z}}(\boldsymbol{z}) \right| = \left| A + \boldsymbol{b} \frac{\partial \psi}{\partial \boldsymbol{z}}(\boldsymbol{z}) \right| \leqslant |A| + |\boldsymbol{b}| L_2. \tag{B-13}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

B.4 Proof of Proposition 5.3

The idea of the proof is to compare $\hat{y}_\theta$ obtained from ψ_θ with the prediction $\hat{y}$ defined in (5-7) obtained employing the nominal mapping ψ. Note that

$$|\psi(z) - \psi_\theta(z_\theta)| \leqslant |\psi(z) - \psi(z_\theta)| + |\psi(z_\theta) - \psi_\theta(z_\theta)| \,. \tag{B-14}$$

With (5-20) and knowing that ψ is L_2-Lipschitz

$$|\psi(z) - \psi_\theta(z_\theta)| \leqslant L_2|z - z_\theta| + \delta \,. \tag{B-15}$$

On the other hand, A being Hurwitz, there exist P a positive definite matrix and $\lambda > 0$ such that

$$PA + A^\top P \leqslant -2\lambda P \,. \tag{B-16}$$

For two vectors $(\boldsymbol{u}, \boldsymbol{v})$ in $\mathbb{R}^m$, let us denote $\langle \boldsymbol{u}, \boldsymbol{v}\rangle_P = \boldsymbol{u}^\top P \boldsymbol{v}$ and $\|\boldsymbol{u}\|_P^2 = \boldsymbol{u}^\top P \boldsymbol{u}$. Along the solutions of the system (5-23) and (5-5) with $g(z) = Az + \boldsymbol{b}\psi(z)$ it yields

$$\frac{\partial}{\partial t}\|z - z_\theta\|_P^2 = (z - z_\theta)^\top (PA + A^\top P)(z - z_\theta) + 2\langle z - z_\theta, \boldsymbol{b}(\psi(z) - \psi_\theta(z_\theta))\rangle_P \,. \tag{B-17}$$

Using Cauchy-Schwarz inequality and (B-16)

$$\frac{\partial}{\partial t}\|z - z_\theta\|_P^2 \leqslant -2\lambda\|z - z_\theta\|_P^2 + 2\|z - z_\theta\|_P\|b\|_P|\psi(z) - \psi_\theta(z_\theta)| \tag{B-18}$$

With (B-15) we get

$$\frac{\partial}{\partial t}\|z - z_\theta\|_P^2 \leqslant 2\Big(L_2\|\boldsymbol{b}\|_P - \lambda\Big)\|z - z_\theta\|_P^2 + \delta\|\boldsymbol{b}\|_P\|z - z_\theta\|_P \tag{B-19}$$

which is equivalent to

$$\frac{\partial}{\partial t}\|z - z_\theta\|_P \leqslant \Big(L_2\|\boldsymbol{b}\|_P - \lambda\Big)\|z - z_\theta\|_P + \delta\|\boldsymbol{b}\|_P \tag{B-20}$$

With Grönwall inequality, it yields,

$$\|\mathcal{Z}(z,p) - \mathcal{Z}_\theta(z,p)\|_P \leqslant \frac{\delta\|\boldsymbol{b}\|_P}{L_2\|\boldsymbol{b}\|_P - \lambda} e^{(L_2\|\boldsymbol{b}\|_P - \lambda)p} \,, \forall (z,p). \tag{B-21}$$

This implies with $\hat{y}$ defined in (5-7) :

$$|\hat{y}(\ell + p) - \hat{y}_\theta(\ell + p)| \leqslant \delta\left(\frac{L_2\|\boldsymbol{b}\|_P}{L_2\|\boldsymbol{b}\| - \lambda} e^{(L_2\|\boldsymbol{b}\|_P - \lambda)p} + 1\right). \tag{B-22}$$

However,

$$|y(\ell + p) - \hat{y}_\theta(\ell + p)| \leqslant |y(\ell + p) - \hat{y}(\ell + p)| + |\hat{y}(\ell + p) - \hat{y}_\theta(\ell + p)|, \tag{B-23}$$

and employing Proposition 5.1 it finally implies

$$|y(\ell + p) - \hat{y}_\theta(\ell + p)| \leqslant kL_2 e^{-\lambda\ell + L_1 p}|z_0^y| + \delta\left(L_4 e^{\frac{L_2\|b\|_P}{L_4}p} + 1\right), \tag{B-24}$$

where k is obtained from P and $L_1 = \|A\| + L_2\|b\|$ and $L_4 = \frac{L_2\|b\|_P}{L_2\|b\|_P - \lambda}$. This concludes the proof.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Appendix C

Appendix of chapter 6

To simplify the analysis, we assume forward completeness of the trajectories for all times $t \geq t_0, t_0 \in \mathbb{R}$ inside a forward invariant compact set $\mathcal{F} \subset \mathbb{R}^{n_s}$. We also assume that if $\boldsymbol{r}(t) \in \mathcal{R}$ for all $t \geq t_0$, then $\pi(t)$ solving (6-18) is bounded and satisfies $\pi(t) \in \mathcal{F}$ for all $t \geq t_0$.

C.1 Proof of Proposition 6.1

The proof follows the line of results in Giaccagli et al. (2022b) and combines them with ISS-like arguments. For space reasons, we only highlight the main parts. Define the state-error $\tilde{\boldsymbol{s}} := \boldsymbol{s} - \pi$. Its dynamics read as

$$\dot{\tilde{\boldsymbol{s}}} = \varphi(\pi + \tilde{\boldsymbol{s}}, t) - \varphi(\pi, t) - \kappa g(\pi + \tilde{\boldsymbol{s}}) \times \big(\beta(\pi + \tilde{\boldsymbol{s}}, t) - \beta(\pi, t)\big) + g(\pi + \tilde{\boldsymbol{s}})\omega(t)\,. \quad \text{(C-1)}$$

Let $\tilde{\mathcal{S}}(\tilde{s}_0, t, t_0)$ be a solution defined for all $t \geq t_0$ and consider the function

$$\Gamma : [0,1] \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{n_s} \text{ s.t } \begin{cases} \Gamma(1, t_0, t_0) &= \tilde{\mathcal{S}}(\tilde{s}_0, t_0, t_0) \\ \Gamma(0, t_0, t_0) &= 0 \\ \Gamma(x, t_0, t_0) &= \gamma(x) \end{cases} \quad \text{(C-2)}$$

where $\gamma : [0,1] \to \mathbb{R}^{n_s}$ is any C^1 curve and solution to

$$\frac{\partial \Gamma}{\partial t}(x, t, t_0) = \varphi(\Delta, t) - \varphi(\Pi, t) - \kappa g(\Delta)\big(\beta(\Delta, t) - \beta(\Pi, t)\big) + g(\Delta) w(x) \quad \text{(C-3)}$$

with $\Delta = \Gamma + \Pi$ and $\Pi = \Pi(\pi_0, t, t_0)$ being the trajectory of (6-18) (arguments are dropped for space reasons) and $w(x) = x\omega$. Take the candidate Lyapunov function

$$V(t) = \int_0^1 \frac{\partial \Gamma}{\partial x}^{\top}(x, t, t_0) P(\Delta, t) \frac{\partial \Gamma}{\partial x}(x, t, t_0)\, ds \quad \text{(C-4)}$$

with P solving (6-20). Taking its time-derivative and through the Killing vector assumption and the integrability condition (6-21), we get

$$\dot{V}(t) \leqslant \int_0^1 \frac{\partial \Gamma}{\partial x}^{\top}(x, t, t_0) \big[T_1(x, t_0, t) + T_2(x, t_0, t)\big] \frac{\partial \Gamma}{\partial x}(x, t, t_0) + T_3(x, t_0, t)\, dx\,, \quad \text{(C-5)}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

with

$$
\begin{aligned}
T_1(x,t_0,t) &= L_\varphi P(\Delta,t)\\
T_2(x,t_0,t) &= -2\kappa P(\Delta,t)g(\Delta)g^\top(\Delta)P(\Delta,t)\\
T_3(x,t_0,t) &= \frac{\partial \Gamma^\top}{\partial x}(x,t,t_0)P(\Delta,t)g(\Delta,t)\omega(t)+\omega^\top(t)g^\top(\Delta,t)P(\Delta,t)\frac{\partial \Gamma}{\partial x}(x,t,t_0)\,.
\end{aligned}
$$

From the (generalized) inequality of Young: $2ab \leq ca^2 + \frac{b^2}{c}$ for any $c > 0$ with

$$
\begin{aligned}
a &= \tfrac{\partial \Gamma^\top}{\partial x}(x,t,t_0)\sqrt{P(\Delta,t)},\\
b &= \sqrt{P(\Delta,t)}g(\Delta)\omega(t)\\
c &= \tfrac{\lambda}{2}
\end{aligned}
$$

it follows that

$$
T_3(x,t,t_0) \leqslant \frac{\lambda}{2}\frac{\partial \Gamma^\top}{\partial x}(x,t,t_0)P(\Delta,t)\frac{\partial \Gamma}{\partial x}(x,t,t_0) + \frac{2}{\lambda}\omega^\top(t)g^\top(\Delta)P(\Delta,t)g(\Delta)\omega(t) \tag{C-6}
$$

Taking $\kappa \geqslant \frac{\rho}{2}$ and $\overline{g} := \sup_{s\in\mathcal{F}} |g(s)|$, employing (6-20) we get

$$
\dot{V}(t) \leqslant -\tfrac{\lambda}{2}V(t) + \tfrac{2}{\lambda}\overline{p}\,\overline{g}^2|\omega(t)|^2\,. \tag{C-7}
$$

From (6-20) and since $\tilde{\mathcal{S}}(\tilde{s},t,t) = \tilde{s}(t)\,\forall\, t$, it follows that, for any $t \geq t_0$

$$
\underline{p}\,|\tilde{s}(t)|^2 \leqslant V(t) \leqslant \overline{p}\,|\tilde{s}(t)|^2\,. \tag{C-8}
$$

Hence, the proof concludes by Gronwall lemma and by following standard ISS-like arguments Sontag and Wang (1995).

C.2 Proof of Proposition 6.2

By adding and subtracting $\psi(t), \beta(s,t), \beta(\pi(t),t)$ and $\beta(\pi_\theta(t),t)$, we rewrite the control (6-25) as $u(t) = u^\star(t) + \tilde{u}(t)$ with $u^\star := \psi(t) - \kappa\big(\beta(s,t) - \beta(\pi(t),t)\big)$ and $\tilde{u}$ defined as

$$
\begin{aligned}
\tilde{u}(t) := \psi_\theta(t) - \psi(t) - \kappa\Big[&\big(\beta_\theta(s,t) - \beta(s,t)\big) + \big(\beta(\pi_\theta(t),t) - \beta_\theta(\pi_\theta(t),t)\big) + \\
&\big(\beta(\pi(t),t) - \beta(\pi_\theta(t),t)\big)\Big]. \qquad \text{(C-9)}
\end{aligned}
$$

Consider the Lyapunov function (C-4) with $\tilde{s} = s - \pi$. Following the same lines as in the proof of Proposition 6.1, it follows that

$$
\dot{V}(t) \leqslant -\tfrac{\lambda}{2}V(t) + \tfrac{2}{\lambda}\overline{p}\,\overline{g}^2|\tilde{u}(t)|^2\,.
$$

Consider now the reference r. Since $r(t) \in \mathcal{R}$ for all $t \geq t_0$, there exist a compact set $\mathcal{W}_\pi$ such that $\pi(t) \in \mathcal{W}_\pi$ for all $t \geq t_0$. Now, define

$$
\eta_1 := \sup_{s\in\mathcal{W}_\pi} |s|\,, \quad \eta_2 := \sup_{s\in\mathcal{W}_{\tilde{s}}} |s|\,, \quad \eta_3 := \max\{\eta_2, \delta\}\,.
$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Then, $\mathcal{W}_{\tilde{s}} \subseteq \mathcal{B}_{\eta_3}$. Define $\overline{\mathcal{V}} := \{\tilde{s}_0 \in \mathbb{R}^n : V(t_0) \leq \bar{p}\eta_3^2\}$, and note that $\tilde{s}_0 \in \mathcal{B}_{\eta_3}$ implies $\tilde{s}_0 \in \overline{\mathcal{V}}$ due to (C-8). Differently put, $\mathcal{B}_{\eta_3} \subseteq \overline{\mathcal{V}}$. Pick $\mathcal{W}_s = \mathcal{B}_{\eta_4}$, where

$$\eta_4 = \eta_1 + \sup_{s \in \overline{\mathcal{V}}} |s| \, .$$

Then, $\overline{\mathcal{V}} \subseteq \mathcal{W}_s$. Moreover, if $\tilde{s}_0 \in \overline{\mathcal{V}}$, then $s_0 \in \mathcal{W}_x$. Let $\overline{g} := \sup_{s \in \mathcal{F}} |g(s)|$ and pick

$$\mu_\delta = \frac{\lambda \delta \underline{p}}{2\sqrt{2}\,\overline{p}\,\overline{g}(1 + 2\kappa + \kappa \overline{p}\,\overline{g})} \, .$$

From (6-26), (C-9) and the relation (6-21), it follows that, for all times $t \geq t_0$ such that $s(t) \in \mathcal{W}_s$

$$|\tilde{u}(t)| \leqslant \mu_\delta + 2\kappa\mu_\delta + \kappa \overline{p}\,\overline{g}\,|\pi(t) - \pi_\theta(t)| \tag{C-10}$$

$$\leqslant \mu_\delta(1 + 2\kappa + \kappa \overline{p}\,\overline{g}) \leqslant \frac{\lambda \delta \underline{p}}{2\sqrt{2}\,\overline{p}\,\overline{g}} \, . \tag{C-11}$$

Consider now the set $\underline{\mathcal{V}} := \{\tilde{s}_0 \in \mathbb{R}^{n_s} : V(t_0) < \underline{p}\delta^2\}$ and suppose $\tilde{s}_0 \in \underline{\mathcal{V}}$. By (C-8), if $\underline{p}\,|\tilde{s}_0|^2 < \underline{p}\delta^2$ then $|s_0 - \pi_0| < \delta$. Now, note that $\underline{\mathcal{V}} \subsetneq \mathcal{B}_{\eta_3} \subseteq \overline{\mathcal{V}}$ and suppose $\tilde{s}_0 \in \overline{\mathcal{V}} \setminus \underline{\mathcal{V}}$. This implies $|\tilde{s}_0|^2 \geq \frac{p}{\bar{p}}\delta^2$. However, since $\tilde{s}_0 \in \overline{\mathcal{V}}$ implies $s_0 \in \mathcal{W}_s$, we have

$$\dot{V}(t_0) \leqslant -\lambda V(t_0) + \tfrac{2}{\lambda}\overline{p}\,\overline{g}^2 |\tilde{u}(t_0)|^2 \tag{C-12}$$

$$\leqslant \tfrac{\lambda}{2}(\tfrac{p^2}{2\bar{p}}\delta^2 - V(t_0)) \tag{C-13}$$

$$\leqslant \underline{p}\tfrac{\lambda}{2}(\tfrac{1}{2}\tfrac{p}{\bar{p}}\delta^2 - |\tilde{s}_0|^2) < 0 \, . \tag{C-14}$$

Hence, the level set $\overline{\mathcal{V}}$ is forward invariant. Moreover, given the above results, the set $\underline{\mathcal{V}}$ is attractive and forward invariant, with the domain of attraction including $\overline{\mathcal{V}}$. Recall that $\mathcal{W}_{\tilde{s}} \subseteq \mathcal{B}_{\eta_3} \subseteq \overline{\mathcal{V}}$. Hence, for all $\tilde{s}_0 \in \mathcal{W}_{\tilde{s}}$, it holds $\lim_{t \to +\infty} |\mathcal{S}(s_0, t, t_0) - \Pi(\pi_0, t, t_0)| \leqslant \delta$.

C.3 Model details

The steady-state reference generator uses two MLPs $p_{1,\theta}, p_{2,\theta}$ with four layers, 64 hidden units and tanh-activated, using layer normalization on intermediate layers. The recurrent networks $c_{1,\theta}, c_{2,\theta}$ are two single-layered GRU with a latent memory of dimension 64. The model is trained with Adam optimizer, with the learning rate set at 10^{-3}, batch size of 256 and 1,000 epochs. Both β_θ and P_θ are modeled with four-layered, 64 hidden units and tanh-activated MLPs. During training, we choose $\omega_i = 1$ for all $i = 1...8$. These models are trained using Adam optimizer with a learning rate of 3×10^{-3} during 100 epochs.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Appendix D

Appendix of chapter 7

D.1 Proof of proposition 7.1

The proof proceeds by successive majorations and triangular inequalities. For sake of clarity, and only in this proof we omit the d subscript and write $s[n]$ and $z[n]$ for $s_d[n]$ and $z_d[n]$, respectively.

We start with $\hat{s}[n] := \hat{h_1} \circ \hat{f_1}^n \circ \hat{e}(s[0])$. Thus for any integer $n > 0$, we have

$$|s[n] - \hat{s}[n]| = |h_1(z[n]) - \hat{h_1}(\hat{z}[n])|. \tag{D-1}$$

Using Lipschitz property and 7-4, then

$$\begin{aligned} |s[n] - \hat{s}[n]| &\leqslant |h_1(z[n]) - \hat{h_1}(z[n])| + |\hat{h_1}(z[n]) - \hat{h_1}(\hat{z}[n])| \\ &\leqslant \delta_h + L_h|z[n] - \hat{z}[n]|. \end{aligned} \tag{D-2}$$

Noticing that one can rewrite $\hat{z}[n]$ as $\hat{z}[n] = \hat{f_1}^n \circ \hat{e}(s[0])$. Since $z[n] = {f_1}^n(z[0])$ and using a similar decomposition as for D-2), one gets:

$$\left|z[n] - \hat{z}[n]\right| \leqslant \delta_f \sum_{k=0}^{n-1} L_f^k + L_f^n \left|z[0] - \hat{z}[0]\right|. \tag{D-3}$$

Hence, from (D-2), and using $z[0] = e(s[0])$ and $\hat{z}[0] = \hat{e}(s[0])$, we have

$$\left|s[n] - \hat{s}[n]\right| \leqslant \delta_h + L_h \left(\delta_f \frac{L_f^n - 1}{L_f - 1} + L_f^n \delta_e\right). \tag{D-4}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

We now move on to the classic auto-regressive case, i.e. $\hat{s}^{\text{ar}}[n] = \left(\hat{h}_1 \circ \hat{f}_1 \circ \hat{e}\right)^n (\boldsymbol{s}[0])$.

$$\begin{aligned}
|\boldsymbol{s}[n] - \hat{\boldsymbol{s}}^{\text{ar}}[n]| &\leqslant \left|h_1(\boldsymbol{z}[n]) - \hat{h}_1(\boldsymbol{z}[n])\right| + \left|\hat{h}_1(\boldsymbol{z}[n]) - \hat{h}_1(\hat{\boldsymbol{z}}^{\text{ar}}[n])\right| \\
&\leqslant \delta_h + L_h|\boldsymbol{z}[n] - \hat{\boldsymbol{z}}^{\text{ar}}[n]| \\
&\leqslant \delta_h + L_h\left(\delta_f + L_f\left|e(\boldsymbol{s}[n-1]) - \hat{e}(\hat{\boldsymbol{s}}^{\text{ar}}[n-1])\right|\right) \\
&\leqslant \delta_h + L_h\left(\delta_f + L_f\left(\delta_e + L_e|\boldsymbol{s}[n-1] - \hat{\boldsymbol{s}}^{\text{ar}}[n-1]|\right)\right) \\
&\leqslant \delta \sum_{i=0}^{n-2} L^i + L^{n-1}|\boldsymbol{s}[1] - \hat{\boldsymbol{s}}^{\text{ar}}[1]|,
\end{aligned} \tag{D-5}$$

with $\delta = \delta_h + L_h\delta_f + L_hL_f\delta_e$ and $L = L_hL_fL_e$. Moreover,

$$\begin{aligned}
|\boldsymbol{s}[1] - \hat{\boldsymbol{s}}^{\text{ar}}[1]| &= |\hat{h}_1(\boldsymbol{z}[1]) - \hat{f}_1(\hat{\boldsymbol{z}}^{\text{ar}}[1])| && \text{(D-6)} \\
&\leqslant \delta_h + L_h|\boldsymbol{z}[1] - \hat{\boldsymbol{z}}^{\text{ar}}[1]| && \text{(D-7)} \\
&\leqslant \delta_h + L_h(\delta_f + L_f|\boldsymbol{z}[0] - \hat{\boldsymbol{z}}^{\text{ar}}[0]|) && \text{(D-8)} \\
&\leqslant \delta && \text{(D-9)}
\end{aligned}$$

Putting it all together, we get equation 7-6:

$$|\boldsymbol{s}[n] - \hat{\boldsymbol{s}}^{\text{ar}}[n]| \leqslant \delta\frac{L^n - 1}{L - 1} \tag{D-10}$$

Finally, (D-4) and (D-10) conclude the proof.

D.2 Comparison of upper bounds in Proposition 7.1

We start by formulating (7-5) and (7-6) under a comparable form

$$|\boldsymbol{s}_d[n] - \hat{\boldsymbol{s}}_d[n]| \leqslant \delta + L_hL_f\delta_f\frac{L_f^{n-1} - 1}{L_f - 1} + L_hL_f\delta_e(L_f^{n-1} - 1) \qquad = \delta + K_1 \tag{D-11}$$

$$|\boldsymbol{s}_d[n] - \hat{\boldsymbol{s}}_d^{\text{ar}}[n]| \leqslant \delta + \delta\frac{L^n - L}{L - 1} \qquad = \delta + K_2 \tag{D-12}$$

Now we consider two cases depending on the Lipschitz constants of the problem, namely L_h, L_f, and L_e. First, consider the case where the Lipschitz constants are very large (i.e. $L_h, L_f, L_e \gg 1$). In that case, the upper bounds can be approached by

$$K_1 \approx L_h\delta_f L_f^{n-1} + L_hL_f^n\delta_e \tag{D-13}$$

$$K_2 \approx \delta_h(L_hL_fL_e)^{n-1} + L_h\delta_f L_f^{n-1}L_h^{n-1}L_e^{n-1} + L_hL_f^n\delta_e L_h^{n-1}L_e^{n-1} \tag{D-14}$$

Hence, $K_2 \gg K_1$ (we highlighted the difference between both terms in the previous equation. Now consider the case where the Lipschitz constants are very small (i.e. $L_h, L_f, L_e \ll 1$). Recall that this case corresponds to a trivial prediction task since any trajectory of System 1 will converge to a unique state. Again, the upper bounds can be approached by

$$K_1 \approx 0 \tag{D-15}$$

$$K_2 \approx L\delta \tag{D-16}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

In this trivial case, the upper bound on the prediction error using our method is a combination of the approximation errors from each function. On the other hand, using the classic AR scheme implies a larger error, since the model accumulates approximations at each time step not only from the dynamics but also from the observation function and the encoder.

D.3 Proof of proposition 7.2

The proof follows the lines of Janny et al. (2022). The existence of ψ_q is granted by the observability assumption. Indeed assumption A2. states that for all $q > p$, $\mathcal{O}_q$ is injective in $\mathcal{S}$. Hence, it exists a inverse mapping $\mathcal{O}_q^* : \mathcal{O}_q : \mathcal{S} \mapsto \mathcal{S}$ such that $\forall \boldsymbol{s}' \in \mathcal{S}$

$$\mathcal{O}_q^*\big(\mathcal{O}_q(\boldsymbol{s}')\big) = \boldsymbol{s}' \tag{D-17}$$

Let $\boldsymbol{z}_d[0|q] = \big[\boldsymbol{z}_d[0] \ \cdots \ \boldsymbol{z}_d[q]\big]$. Hence, one can build ψ_q using the dynamics of the system for all $\boldsymbol{x} \in \Omega$:

$$\forall \boldsymbol{s}_0 \in \mathcal{S}, \quad S(\boldsymbol{s}_0, \boldsymbol{x}, t) = S\Big(\mathcal{O}_q^*\big(\boldsymbol{z}_d[0|q]\big), \boldsymbol{x}, t\Big) := \psi_q\big(\boldsymbol{z}_d[0|q], \boldsymbol{x}, t\big) \tag{D-18}$$

Now, because of the noise, the disturbed observation $\hat{\boldsymbol{z}}_d[0|q] = \boldsymbol{z}_d[0|q] + \delta_{0|q}$ may not belong to $\mathcal{O}_q(\mathcal{S})$, where the inverse mapping $\mathcal{O}_q^*$ is well defined. We solve this by finding the closest "possible" observation.

$$\hat{\boldsymbol{s}}_0 = \arg\min_{\boldsymbol{s}' \in \mathcal{S}} \big|\hat{\boldsymbol{z}}_d[0|q] - \mathcal{O}_q(\boldsymbol{s}')\big| \tag{D-19}$$

$$\hat{\boldsymbol{s}}(\boldsymbol{x}, t) = S(\hat{\boldsymbol{s}}_0, \boldsymbol{x}, t) := \psi_q\big(\hat{\boldsymbol{z}}_d[0|q], \boldsymbol{x}, t\big). \tag{D-20}$$

Hence, we have for all $\boldsymbol{s}' \in \mathcal{S}$

$$\Big|\hat{\boldsymbol{z}}_d[0|q] - \mathcal{O}_q\big(\hat{\boldsymbol{s}}_0\big)\Big| \leqslant \Big|\hat{\boldsymbol{z}}_d[0|q] - \mathcal{O}_q\big(\boldsymbol{s}'\big)\Big|. \tag{D-21}$$

In particular, for $\boldsymbol{s}' = \boldsymbol{s}_0$ and since $\mathcal{O}_q(\boldsymbol{s}_0) = \boldsymbol{z}_d[0|q]$,

$$\begin{aligned}\Big|\hat{\boldsymbol{z}}_d[0|q] - \mathcal{O}_q\big(\hat{\boldsymbol{s}}_0\big)\Big| &\leqslant \Big|\hat{\boldsymbol{z}}_d[0|q] - \mathcal{O}_q\big(\boldsymbol{s}_0\big)\Big| \\ &\leqslant |\delta_{0|q}|.\end{aligned} \tag{D-22}$$

In the other hand, from assumption A2. (7-8):

$$\begin{aligned}\alpha(p)|\hat{\boldsymbol{s}}_0 - \boldsymbol{s}_0|_{\mathcal{S}} &\leqslant |\mathcal{O}_q(\hat{\boldsymbol{s}}_0) - \mathcal{O}_q(\boldsymbol{s}_0)| \\ &\leqslant |\mathcal{O}_q(\hat{\boldsymbol{s}}_0) - \hat{\boldsymbol{z}}_d[0|q]| + |\hat{\boldsymbol{z}}_d[0|q] - \mathcal{O}_q(\boldsymbol{s}_0)| \\ &\leqslant 2\big|\delta_{0|q}\big|\end{aligned} \tag{D-23}$$

Moreover, since f_2 is Lipschitz

$$\begin{aligned}\frac{\partial}{\partial t}|S(\boldsymbol{s}_0, \boldsymbol{x}, t) - S(\hat{\boldsymbol{s}}_0, \boldsymbol{x}, t)|_{\mathcal{S}} &= |f_2\big(S(\boldsymbol{s}_0, \boldsymbol{x}, t)\big) - f_2\big(S(\hat{\boldsymbol{s}}_0, \boldsymbol{x}, t)\big)|_{\mathcal{S}} \\ &\leqslant L_s|S(\boldsymbol{s}_0, \boldsymbol{x}, t) - S(\hat{\boldsymbol{s}}_0, \boldsymbol{x}, t)|_S.\end{aligned} \tag{D-24}$$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

and using the Grönwall inequality

$$|S(\boldsymbol{s}_0, \boldsymbol{x}, t) - S(\hat{\boldsymbol{s}}_0, \boldsymbol{x}, t)|_{\mathcal{S}} \leqslant e^{L_s t} |\boldsymbol{s}_0 - \hat{\boldsymbol{s}}_0|_{\mathcal{S}}. \tag{D-25}$$

Finally, combining (D-23) and (D-25)

$$|S(\boldsymbol{s}_0, \boldsymbol{x}, t) - S(\hat{\boldsymbol{s}}_0, \boldsymbol{x}, t)|_{\mathcal{S}} \leqslant 2\alpha(q)^{-1} |\delta_{0|q}| e^{L_s t}.$$

which concludes the proof.

D.4 Model description

In this section, we describe the architecture of our implementation in more detail.

Step 1 – The output predictor derived from System 1 is implemented as a multi-layer graph neural network inspired from Pfaff et al. (2020); Sanchez-Gonzalez et al. (2020) but without following the standard "*encode-process-decode*" setup. Let $\tilde{\mathcal{X}} = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_K\}$ be the set of sub-sampled positions extracted from the known locations $\mathcal{X}$ (cf. Artificial generalization from section 7.2.4). The input of the module is the initial condition at the sampled points and the corresponding positions $\left(\boldsymbol{x}_i, \tilde{\boldsymbol{s}}_d[0](\boldsymbol{x}_i)\right)_i$ and is encoded into a graph-structured latent space $\boldsymbol{z}_d[0] = (\boldsymbol{z}_d[0]_i, \boldsymbol{e}[0]_{ij})_{i,j}$ where $\boldsymbol{z}_d[0]_i$ is a latent node embedding for position $\boldsymbol{x}_i$ and $\boldsymbol{e}[0]_{ij}$ is an edge embedding for edge pairs (i, j) extracted from a Delaunay triangulation. The encoder $\hat{e}$ maps the sparse IC to node and edge embeddings using two MLPs, f_{edge} and f_{node}:

$$\boldsymbol{z}_d[0]_i = f_{\text{node}}\left(\tilde{\boldsymbol{s}}_d[0](\boldsymbol{x}_i), \boldsymbol{x}_i\right), \quad \boldsymbol{e}_[0]ij = f_{\text{edge}}\left(\boldsymbol{x}_i - \boldsymbol{x}_j, |\boldsymbol{x}_i - \boldsymbol{x}_j|\right), \tag{D-26}$$

f_{nodes} and f_{edges} are two ReLU-activated MLPs, each consisting of 2 layers with 128 neurons. The initial node and edge features $\boldsymbol{z}_d[0]_i$ and $\boldsymbol{e}[0]_{ij}$ are represented as 128-dimensional vectors.

The dynamics $\hat{f}_1$ is modeled as a multi-layered graph neural network inspired from Pfaff et al. (2020); Sanchez-Gonzalez et al. (2020), we therefore add a layer superscript $^\ell$ to the notation:

$$\boldsymbol{z}_d[n+1] = \hat{f}_1\left(\boldsymbol{z}_d[n]\right) = \left(\boldsymbol{z}_i^L, \boldsymbol{e}_{ij}^L\right)_{i,j} \text{ such that } \begin{cases} \boldsymbol{e}_{ij}^{\ell+1} & = \boldsymbol{e}_{ij}^\ell + \overbrace{g_{\text{edge}}^\ell\left(\boldsymbol{z}_i^\ell, \boldsymbol{z}_j^\ell, \boldsymbol{e}_{ij}^\ell\right)}^{\varepsilon_{ij}}, \\ \boldsymbol{z}_i^{\ell+1} & = \boldsymbol{z}_i^\ell + g_{\text{node}}^\ell\left(\boldsymbol{z}_i^\ell, \sum_j \varepsilon_{ij}\right), \\ \boldsymbol{e}_{ij}^0 & = \boldsymbol{e}[n]_{ij}, \\ \boldsymbol{z}_i^0 & = \boldsymbol{z}_d[n]_i, \end{cases} \tag{D-27}$$

The GNNs employ two MLPs g_{node}^ℓ and g_{edges}^ℓ with same dimensions as f_{edges} and f_{nodes}. We compute the sequence of anchor states $\boldsymbol{z}_d[0], \cdots \boldsymbol{z}_d[q]$ in the latent space by applying $\hat{f}_1$ auto-regressively.

The observation function $\hat{h}_1$ extracts the sparse observations $\tilde{\boldsymbol{s}}_d[n]$ from the latent state $\boldsymbol{z}_d[n]$ and consists of a two-layered MLP with 128 neurons, with Swish activation functions (Ramachandran et al., 2017) applied on the node features, i.e. $\tilde{\boldsymbol{s}}_d[n](\boldsymbol{x}_i) \approx \hat{h}_1\left(\boldsymbol{z}[n]_i\right)$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Step 2 – The spatial and temporal domains $\Omega \times [\![0, T]\!]$ are normalized, since it tends to improve generalization on unseen locations. The state estimator ψ_q takes as input the sequence of latent graph representation $z_d[0], \cdots, z_d[q]$ and a spatiotemporal query sampled in $\Omega \times [\![0, T]\!]$. This query is embedded in a Fourier space using the function ζ_ω which depends on a frequency parameter $\omega \in \mathbb{R}^{\dim \Omega + 1}$ (initialized uniformly in $[0, 1]$). By concatenating harmonics of this frequency up to some rank, we obtain a resulting embedding of 128 dimensions (if $\zeta_\omega(\boldsymbol{x}, t)$ exceeds the number of dimensions, cropping is performed to match the target shape).

$$\zeta_\omega(\boldsymbol{x}, t) = [..., \cos(k\omega_{1|n_x}\mathbf{x}), \sin(k\omega_{1|n_x}\boldsymbol{x}), \cos(k\omega_{n_x+1}t), \sin(k\omega_{n_x+1}t), ...], \; k \in \{0, \cdots K\}. \tag{D-28}$$

The continuous variables $z_{n\Delta}(\boldsymbol{x}, t)$ conditioned by the anchor states are computed with multi-head attention Vaswani et al. (2017)

$$z_{n\Delta}(\boldsymbol{x}, t) = f_{\text{mha}}\big(\text{Q}{=}\zeta_\omega(\boldsymbol{x}, t), \text{K}{=}\text{V}{=}\{z_d[n]_i\} + \zeta_\omega(\mathcal{X}, n\Delta)\big), \tag{D-29}$$

where f_{mha} is defined as

$$\begin{cases} q_1 & = A(Q, K, V), \\ q_2 & = Q + q_1, \\ q_3 & = B(q_2), \\ \text{out} & = q_3 + q_2. \end{cases} \tag{D-30}$$

Here, $A(\cdot, \cdot, \cdot)$ refers to the multi-head attention mechanism described in (Vaswani et al., 2017) with four attention heads, and $B(\cdot)$ represents a single-layer multi-layer perceptron activated by the rectified linear unit (ReLU) function. We do not use layer normalization.

The Gated Recurrent Unit Cho et al. (2014) aggregates the sequence of conditioned variables (of length q) as follows:

$$\boldsymbol{u}[n] = r_{\text{gru}}\big(\boldsymbol{u}[n{-}1], z_{n\Delta}(\boldsymbol{x}, t)\big), \tag{D-31}$$

$$\hat{S}(\boldsymbol{s}_0, \boldsymbol{x}, t) = D\left(\boldsymbol{u}[q]\right), \tag{D-32}$$

where $\boldsymbol{u}[n]$ is the hidden memory of a GRU, initialized at zero. r_{GRU} denotes the update equations of a GRU – we omit gating functions from the notation – and D is a decoder MLP that maps the final GRU hidden state to the desired output, that is, the value of the solution at the desired spatio-temporal coordinate $(\boldsymbol{x}, t)$, We used a two-layered gated recurrent unit with a hidden vector of size 128, and a two-layered MLP with 128 neurons activated by the Swish function for D.

Training loop – To create artificial generalization scenarios during training, we employ spatial sub-sampling. Specifically, during each gradient iteration, we randomly and uniformly mask 25% of $\mathcal{X}$ and feed the remaining 75% to the output predictor (System 1). To reduce training time further and improve generalization on unseen locations, we use bootstrapping by randomly sampling a smaller set of points for querying the model (i.e. as inputs to ψ_q). To do so, we maintain a probability weight vector W of dimension $|\mathcal{X} \times \mathcal{T}|$, initialized to one. At

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

each gradient descent step, we randomly select N=1,024 points from $\mathcal{X} \times \mathcal{T}$ weighted by W. We update the weight matrix by setting the values at the sampled locations to zero and then adding the loss function value to the entire vector. This procedure serves two purposes: (a) it keeps track of poorly performing points (with higher loss) and (b) it increases the sampling probability for points that have been infrequently selected in previous steps.

The choice of Δ in the dynamics loss (7-13) allows us to reduce the complexity of the model. In Table 7.1, we present results obtained with $\Delta = 3\Delta^*$ indicating that the output predictor (System 1) predicts the latent state representation three time steps later. Consequently, the number of auto-regressive steps during training decreases from T/Δ^* (e.g., for MeshGraphNet and MAgNet) to T/Δ. In Table 7.2, we used $\Delta = 2\Delta^*$. For a more comprehensive discussion on the effect of Δ on performances, please refer to Appendix D.6.

Training parameters – To be consistent, we trained our model with the same training setup over all different experiments (i.e. same loss function, and same hyper-parameters). However, for the baseline experiments, we did adapt hyper-parameters and used the ones provided by the original authors when possible (see further below). We used the AdamW optimizer with an initial learning rate of 10^{-3}. Models were trained for 4,500 epochs, with a scheduled learning rate decay multiplied by 0.5 after 2,500; 3,000; 3,500; and 4,000 epochs. Applying gradient clipping to a value of 1 effectively prevented catastrophic spiking during training. The batch size was set to 16.

D.5 Baselines and datasets details

D.5.1 *Baselines*

The baselines are trained with the AdamW optimizer with a learning rate set at 10^{-3} for 10,000 epochs on each dataset. We keep the best-performing parameters on the validation set for evaluation on the test set.

DINo – we used the official implementation and kept the hyper-parameters suggested by the authors for *Navier* and *Shallow Water*. For Eagle, we used the same hyper-parameters as for *Shallow Water*. The training procedure was left unchanged.

MeshGraphNet – we used our own implementation of the model in PyTorch, with 8 layers of GNNs for *Navier* and *Shallow Water*, and up to 15 for *Eagle*. Other hyper-parameters were kept unchanged. We warmed up the model with single-step auto-regressive training with noise injection (Gaussian noise with a standard deviation of 10^{-4}), as suggested in the original paper, and then fine-tuned the parameters by training on the complete available horizon. Both steps try to minimize the mean squared error between the prediction and the ground truth. Edges are computed using Delaunay triangulation. During evaluation, we perform cubic interpolation between time steps (linear interpolation gives better results on Eagle) first,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

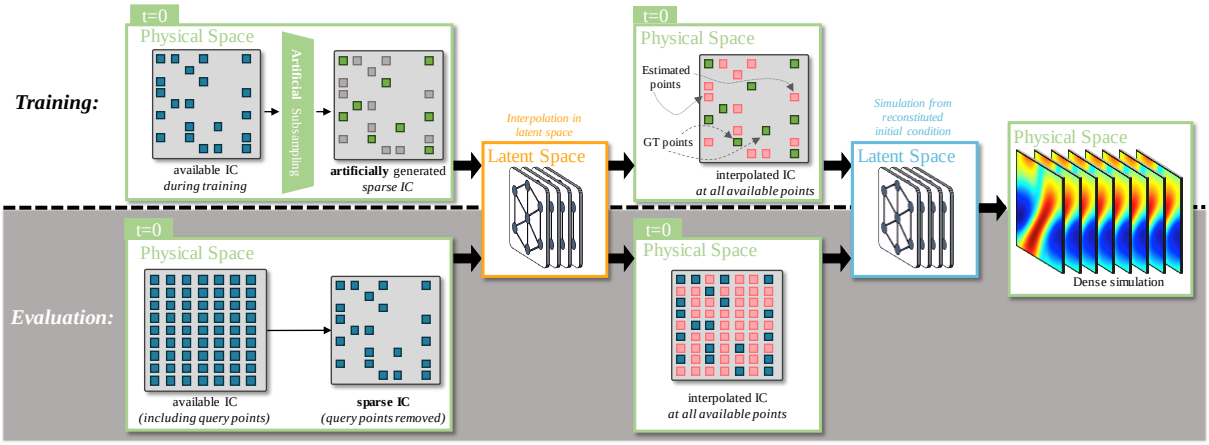


Figure D.1: **MaGNet** – suffers from drastic shifts in distribution between training and evaluation. The model is trained on points from $\mathcal{X}$, which corresponds to a small portion of the domain. We used our subsampling trick to artificially generate queries. During evaluation, we require the prediction at every available point in the complete simulation, hence, MaGNet must interpolate the initial condition to a large number of query points, filling the input of the auto-regressive model with noisy estimates of the IC.

then 2D cubic interpolation on space to retrieve the complete mesh.

MAgNet – We used our own implementation of the MAgNet[GNN] variant of the model, and followed the same training procedure as for MeshGrapNet. The parent mesh and the query points are extracted from the input data using the same spatial sub-sampling technique as ours, and the edges are also computed with Delaunay triangulation. During evaluation, we split the query points into chunks of 10 nodes, and compute their representation with all the available measurement points. This reduces the number of interpolated vertices in the input mesh and improves performances at the cost of higher computation time (see figure D.1). However, to be fair, this increase in computational complexity introduced by ourselves was not taken into account when we discussed computational complexity in appendix D.6.

D.5.2 *Dataset details*

Navier & Shallow Water – Both datasets are derived from the ones used in (Yin et al., 2022). We adopted the same experimental setup but generated distinct training, validation, and testing sets. For details on the GT simulation pipeline, please see Yin et al. (2022). The *Navier* dataset comprises 256 training simulations of 40 frames each, with additional two times 64 simulations allocated for validation and testing. Simulations are conducted on a uniform grid of 64 by 64 pixels (i.e. Ω), measuring the vorticity of a fluid subject to periodic forcing. During training, simulations were cropped to $T = 20$ frames. The *Shallow Water* dataset consists of 64 training simulations, along with 16 simulations in both validation and testing. Sequences of length $T = 20$ were generated. The non-euclidean sampling grid for this dataset is of dimensions 128×64.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

		Navier		
		High	Mid	Low
DINo	In-$\mathcal{X}$	2.266	2.017	3.154
	Ext-$\mathcal{X}$	2.317	2.136	6.740
Interp. MGN	In-$\mathcal{X}$	6.853	3.136	1.378
	Ext-$\mathcal{X}$	7.632	6.890	15.55
MAgNet	In-$\mathcal{X}$	171.5	31.07	10.02
	Ext-$\mathcal{X}$	227.0	57.60	89.20
Ours	In-$\mathcal{X}$	**0.3732**	**0.3563**	**0.3366**
	Ext-$\mathcal{X}$	**0.3766**	**0.3892**	**0.6520**

Table D.1: **Time Extrapolation** – We assessed the performances of our model vs. the baselines in a time-extrapolation scenario by forecasting the solution on a horizon two times longer than the training one (i.e. 40 frames). Our model remains more performant.

Eagle – Eagle is a large-scale fluid dynamics dataset simulating the airflow generated by a drone within a 2D room. We extract sequences of length $T = 10$ from examples within the dataset, limiting the number of points to 3,000 (vertices were duplicated when the number of nodes fell below this threshold).

The spatially down-sampled versions of these datasets (employed in Table 7.1 and 7.2) were obtained through masking. We generate a random binary mask, shared across the training, validation, and test sets, to remove a specified number of points based on the desired scenario. Consequently, the observed locations remain consistent across training, validation, and test sets, except Eagle, where the mesh varies between simulations. For *Navier* and *Shallow Water*, the *High* setup retains 25% of the original grid, the *Middle* setup retains 10%, and the *Low* setup retains 5%. In the case of Eagle, the *High* setup preserves 50% of the original mesh, while the *Low* setup retains only 25%. Temporal down-sampling was also applied by regularly removing a fixed number of frames from the sequences, corresponding to no down-sampling (1/1 setup), half down-sampling (1/2), and quarter down-sampling (1/4). During evaluation, the models are tasked with predicting the solution to every location and time instant present in the original simulation.

D.6 More results

Time continuity – is illustrated in Figure D.3 on the *Navier* dataset. Our model and the baselines are trained in a very challenging setup, where only part of the information is available. During training, not only does the spatial mesh only contains 25% of the complete simulation grid, but also the time-step is increased to four time its initial value. In this situation, the model needs to represent low-resolution data while being trained on sparse data.

Generalization to unseen future timesteps – Beyond time continuity, our model offers some generalization to future timesteps. Table D.1 shows extrapolation results for high/mid/low

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

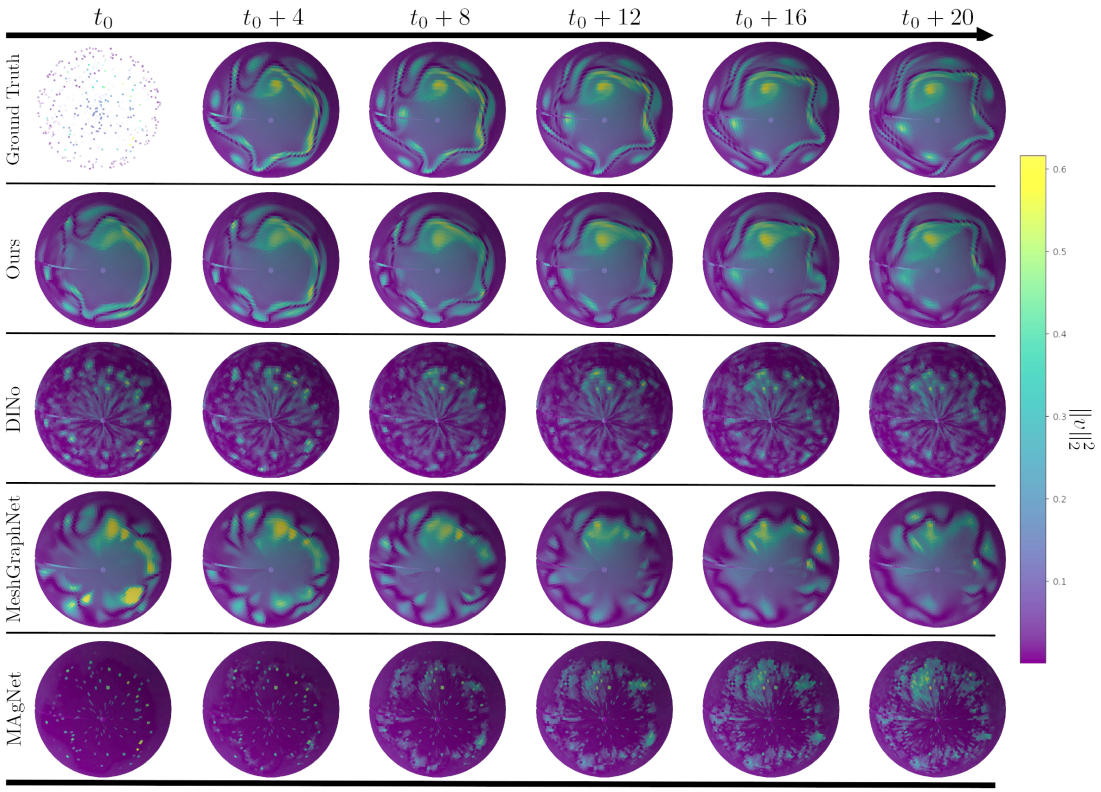


Figure D.2: **Qualitative results on Shallow-Water** – Simulation obtained with our model and the baseline in the challenging 5% setup on the *Shallow Water* dataset (without temporal sub-sampling). Each model is initialized with a small set of sparse observations and needs to extrapolate the solution at many unseen positions. Our model outperforms the baselines, which struggle to compute the solution outside the training domain.

subsampling of the spatial data on the *Navier* dataset which outperforms the predictions of competing baselines.

Generalization to unseen grid – In our spatial and temporal interpolation experiments (tables 7.1 and 7.2 of the chapter), we assumed that the observed mesh remains identical during training and testing. Nevertheless, the ability to adapt to diverse meshes is an important aspect of the task. To evaluate this capability, we trained our model in the spatial extrapolation setup on the Navier dataset. We compute the error when exposed to different meshes, potentially with a different sampling rate, and report the results in table D.2. Our model demonstrates good generalization skills when confronted with new and unseen grids. We observe that the error on new grids is close to the error reported in table 7.1 in the Ext-$\mathcal{X}$ case, we show additionally that the model can generalize even if the observed grid is different. Notably, the model performs well when trained with a medium sampling rate. Despite some performance degradation when the evaluation setup is significantly different compared to training, our model effectively maintains its interpolation quality between out-of-domain error (Ext-$\mathcal{X}$) and in-domain error, testifying to the robustness of our dynamic interpolation module.

Ablations – we study the impact of key design choices in Figure D.4a. First, we show the effect

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

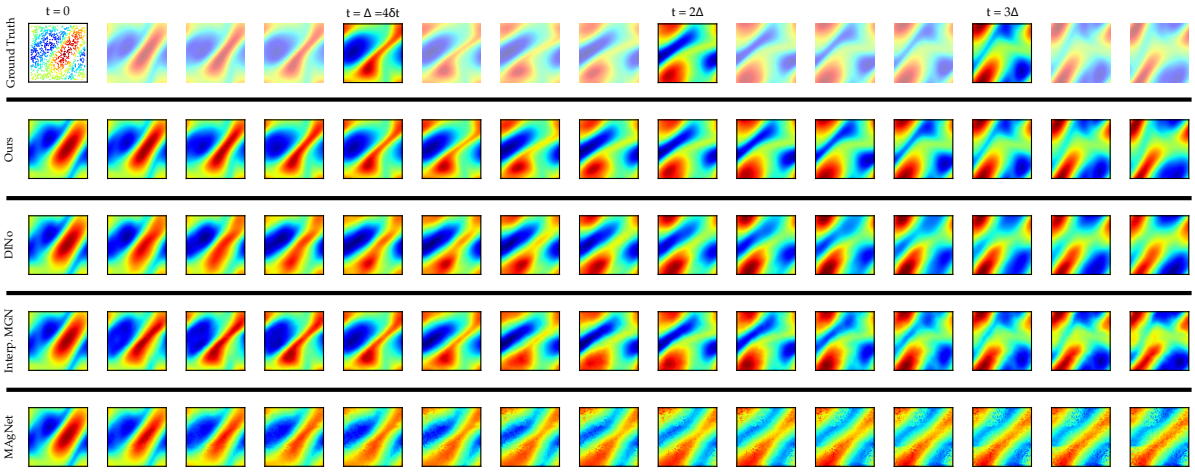


Figure D.3: **Time continuity on the *Navier* dataset** – during training, models are only exposed to a sparse observation of the trajectories, represented spatially by the dots in the upper left figure and temporally by the semi-transparent frames. Our model maintains the temporal coherence of the solution and outperforms the baselines.

of the subsampling strategy to favor learning of spatial generalization, c.f. Section 7.2.4, where we sub-sample the input to the auto-regressive backbone by keeping 75% of the mesh. We ablate this feature by training the model on 100%, 50%, and 25% of the input points. When the model is trained on 100% of the mesh, it fails to generalize to unseen locations, as the model is always queried on points lying in the input mesh. However, reducing the number of input points significantly further from the operating point decreases the performance of the backbone, as it does not dispose of enough points to learn meaningful information for prediction. We also replace the final GRU with simpler aggregation techniques, such as a mean and a maximum pooling, which drastically degrades the results. Finally, we ablate the dynamics part of the training loss (Eq. 7-13). As expected, this deteriorates the results significantly.

More ablation on the interpolator – We conducted an ablation study to show that limiting attention is detrimental. To do so, we designed four variants of our interpolation module:

- **Single attention (w/o GRU)** – performs the attention between the query and the embeddings in a single shot, rather than time-step per time-step. This variant neglects the insights from control theory presented in section 7.2.1 (Step 2). The single softmax function limits the attention to a handful of points, whereas our method encourages the model to attend to at least one point per time step and reason on a larger timescale, considering past and future predictions, which is beneficial for interpolation tasks, as supported by proposition 7.2.
- **Spatial (w/ GRU) & Temporal (w/o GRU) neighborhood** – limit the attention to the nearest temporal or spatial points, which significantly degrades the metrics. To handle setups with sparse and subsampled trajectories, the interpolation module greatly

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

			Training					
			Navier			Shallow		
			High	Mid	Low	High	Mid	Low
Evaluation	High	In-$\mathcal{X}$	**0.2492**	0.7929	4.5165	**0.5224**	1.5431	4.3447
		Ext-$\mathcal{X}$	**0.2477**	0.7782	4.4038	**0.5256**	1.5822	4.4963
	Mid	In-$\mathcal{X}$	0.4370	**0.3230**	0.9759	0.8528	**1.2908**	3.6766
		Ext-$\mathcal{X}$	0.4410	**0.3401**	0.9496	0.8617	**1.2589**	3.6043
	Low	In-$\mathcal{X}$	2.2000	0.4039	**0.6732**	2.4395	1.5634	**3.4793**
		Ext-$\mathcal{X}$	2.2037	0.4216	**0.7892**	2.3914	1.5313	**3.2334**

Table D.2: **Generalization to unseen grid** – We investigate generalization to previously unseen grids by training our model on the Navier dataset in the space extrapolation setup. We report the error (MSE ($\times 10^{-3}$)) inside and outside the spatial domain $\mathcal{X}$ measured with different sampling rates unseen during training. The diagonal shows results on grids with identical sampling rates wrt. training, but sampled differently. Our model shows great generalization properties.

	Ours	Single attention	Temporal attention	Spatial neigh.	Temporal neigh.	ANP Kim et al. (2018)
In-$\mathcal{X}$ / In-$\mathcal{T}$	**0.2113**	0.3863	0.2912	0.5623	0.4130	1.734
Ext-$\mathcal{X}$ / In-$\mathcal{T}$	**0.2251**	0.4168	0.3180	0.6328	0.6681	1.835
In-$\mathcal{X}$ / Ext-$\mathcal{T}$	**0.2235**	0.4094	0.3095	0.6030	1.9624	1.820
Ext-$\mathcal{X}$ / Ext-$\mathcal{T}$	**0.2371**	0.4388	0.3350	0.6741	2.1818	1.920

Table D.3: **Ablation on interpolation** – We performed four ablations on the interpolation module (MSE ($\times 10^{-3}$)). *Single attention* combines all $\boldsymbol{z}_d[n]$ into a single key vector, employing attention only once (w/o GRU). *Temporal attention* replaces the GRU with a 2-head attention, *Spatial neigh.* restricts attention to the five spatially nearest points from the query, and *Temporal neigh.* computes attention only with the nearest time $\boldsymbol{z}_d[n]$ to the queried time τ (w/o GRU). These results indicate that considering long-range spatial and temporal interactions is beneficial for the interpolation task.

benefits from not only distant points but also from the temporal flow of the simulation.

- **Temporal attention (w/o GRU)** replaces the GRU in our model with a 2-head attention layer. This variant of our model does not improve the performance compared to a GRU. We argue that GRU is more suited for accumulating observations in time, as its structure matches classic observer designs in control theory.
- **Attentive Neural Process** Kim et al. (2018) is a interpolation module close to ours resembling the *Single attention* ablation, with an additional global latent $\boldsymbol{c}$ to account for uncertainties. The model involves a prior function $q(\boldsymbol{c}, \boldsymbol{s})$ trained to minimize the Kullback-Leibler divergence between $q\Big(\boldsymbol{z}, \boldsymbol{s}(\mathcal{X}, \mathcal{T})\Big)$ (computed using the physical state at observed points) and $q\Big(\boldsymbol{c}, \boldsymbol{s}(\Omega \setminus \mathcal{X}, [\![0, T]\!])\Big)$ (computed using the physical state at query points).

Results are shown in table D.3. All ablations exhibit worse performance than ours. Note that the ANP ablation involves performing the interpolation in the physical space to compute the Kullback-Leibler divergence during training. Thus, the interpolation module cannot use the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

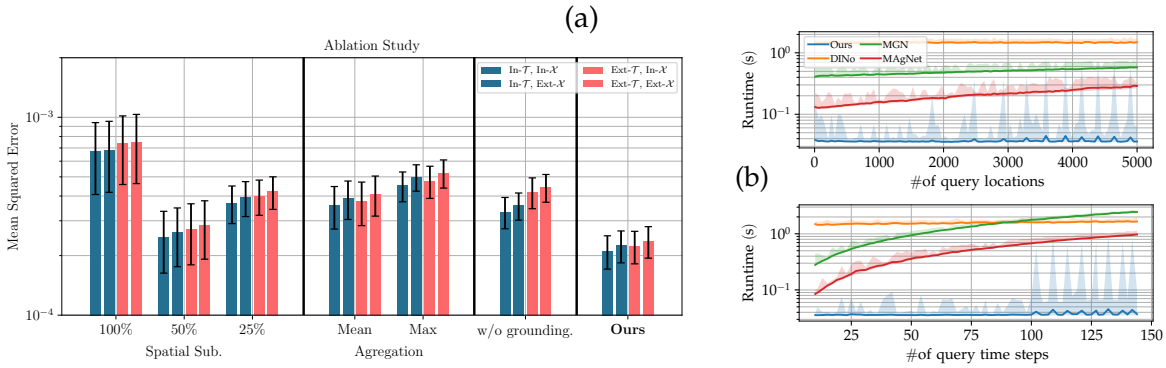


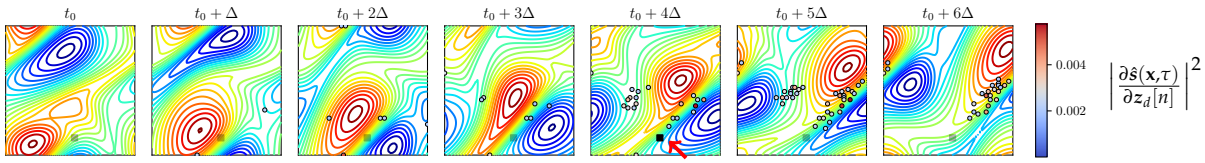
Figure D.4: **Ablations and runtime** – (a) Ablations on Navier (Yin et al., 2022; Stokes, 2009) with 10% of data and half temporal resolution, from left to right: exploring subsampling strategies, replacing GRU par mean/max pooling, removing physics grounding. (b) Runtime analysis as a function of query locations and time steps, respectively.

latent space from the auto-regressive module, which may explain the drop in performance. Adaptating ANP to directly leverage the latent states is probably possible, but not straightforward and requires several key changes in the architecture.

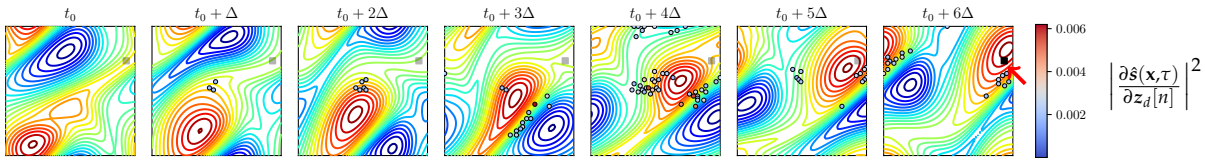
Efficiency – the design choices we made led to a computationally efficient model, compared to prior work. For all three baselines, the required number of computed time steps for the auto-regressive rollout depends on (1) the number of predicted time steps, and (2) the time values themselves, as for later values of t, more iterations need to be computed. In contrast, our method forecasts using attention from a set of "anchor states", which is controlled through the hyper-parameter Δ. The length of the auto-regressive rollout is therefore constant and does not depend on the number of predicted time steps. Furthermore, while DINo scales very well to predict additional locations, it requires a costly optimization step to compute α_0. MGN does benefit from the efficient cubic interpolation algorithm, which is a side effect of the fact that it has been adapted to this task, but not designed for it. We experimentally confirm these claims in Figure D.4, where we provide the evolution of runtime as a function of query locations, and of query time steps, respectively. In both cases, our model compares very favorably to competing methods.

Attention maps – To further support our claims, we analyzed the behavior of the interpolation module in more depth and showed the top-100 most important nodes from the embedding points $z_d[n](x_i)$ used to interpolate at different queries. The figure is shown in Figure D.5. We observed very complex behaviors that dynamically adapt to the global situation around the queried points. Our interpolation module appears to give more importance to the flow rather than merely averaging the neighboring nodes, thus relying on "why" the queried point is in a specific state. Such behavior would be extremely difficult to implement in a handcrafted algorithm.

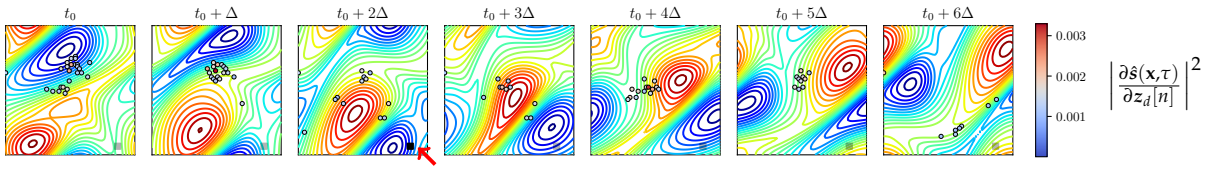
Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) **Frontier tracking**: when queried on a streamline between areas of opposite vorticity, the interpolation module attends not only to the spatial neighbourhood but also to the temporal flow near the frontier.



(b) **Blob tracking**: in homogeneous areas, the model tracks the origin of the perturbation, and focuses on its displacement. Our dynamic interpolation exploits the evolution of the state rather than merely averaging neighboring nodes.



(c) **Periodic boundaries**: our model effectively leverages the periodic condition of the Navier dataset, especially when queried on points originating from perturbations on the other side of the simulation. Again, the interpolation depends on which points explain the output, rather than the neighborhood.

Figure D.5: **Norm of output derivative** – wrt. each $z_d[n](x_i)$ (Navier, high spatial subsampling setup). We display top-100 nodes (•) with the highest norm, i.e. most important nodes for interpolation at query point (■). Using gradients rather than attention allows to visualize the action of the GRU. We observe context-adaptive behaviors, leveraging temporal flow information over local neighbors, challenging to implement in handcrafted algorithms.

Parameter Sensitivity Analysis – We investigate the influence of two principal hyper-parameters, namely the step size Δ and the number of residual GNN layers L, on the performances of our model. We present the results of our experiments in figure D.6 on the *Navier* dataset, which has been spatially down-sampled at 10% during training and has a temporal resolution reduced by two.

The choice of the step size between iterations of the auto-regressive backbone directly affects both training and inference time. For a trajectory of T frames, the number of anchor states $z_d[n]$ is determined by $\lfloor T/\Delta \rfloor$. Increasing the step size Δ of the learned dynamics leads to a higher number of embeddings over which the models need to reason. A parallel can be drawn between this phenomenon and the influence of the discretization size on the accuracy of numerical methods for solving PDEs. Furthermore, the selection of Δ also impacts the generalization capabilities of the model in Ext-$\mathcal{T}$. When $\Delta > \Delta^*$, the model is queried during training on intermediate instants not directly associated with any of the anchor states $z_d[n]$. This is visible in Figure D.6 where, for instance, with $\Delta = \Delta^*$, the In-$\mathcal{X}$/In-$\mathcal{T}$ error is the lowest, but other metrics increases compared to $\Delta = 2\Delta^*$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

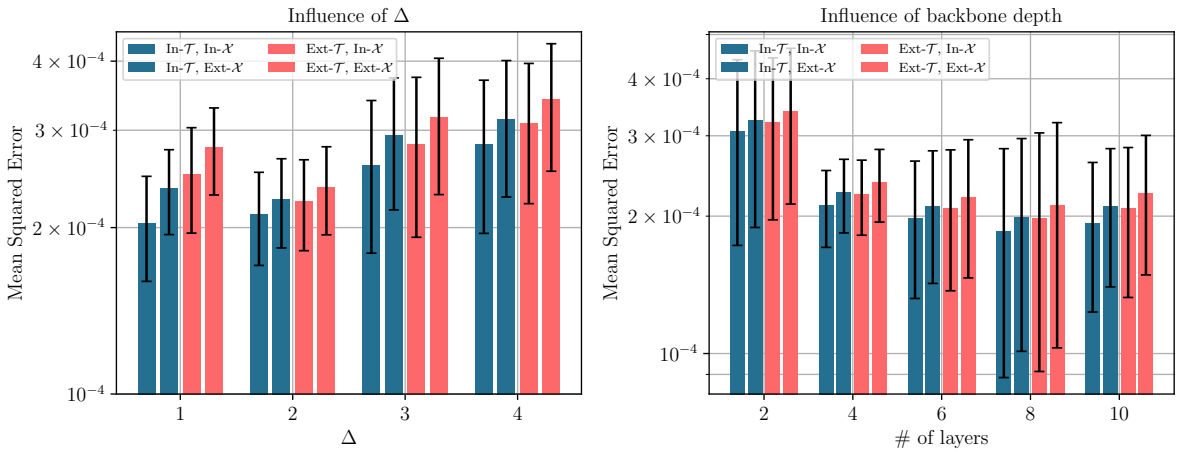


Figure D.6: **Impact of hyper-parameters on model performance** – We evaluate the impact of two critical hyper-parameters on our architecture, namely, the step size Δ and the depth L of the physics backbone. To assess the performance, we employ the 10% *Navier* dataset with 1/2 frames and compute metrics for both in-domain and out-domain. The results reveal that increasing the depth of the GNN layers enhances the model's performance, while lower values of Δ lead to better metrics. However, we observed a degradation in the model ability to generalize to unseen time instants for the special case $\Delta{=}\delta t$.

The number of layers L in the auto-regressive backbone significantly influences the overall performance of the model, both within the domain and on the exteriors. Increasing the number of layers generally leads to improved performance. However, it appears that beyond $L = 8$, the error starts to increase, indicating a saturation point in terms of performance gain. The relationship between the number of layers and model performance is visually depicted in Figure D.6. Throughout this chapter, we maintained this hyper-parameter constant for the sake of simplicity, as our primary focus is the spatial and temporal generalization of the solution.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Appendix E

Appendix for chapter 8

E.1 Dataset details

E.1.1 *Structure and Post-processing*

The *Eagle* dataset is composed of exactly 1,184 simulations of 990 time steps (33 seconds at 30 fps). Scene geometries are arranged in three categories based on the order of the interpolation used to generate the ground structure: 197 **Step** scenes, 199 **Triangular**, and 196 **Spline**. A geometry gives two simulations depending on whether the drone is crossing the left or the right part of the scene. A proper train/valid/test splitting is provided ensuring that each geometry type is equally represented. The train split contains 948 simulations, while test and valid splits each contain 118 simulations.

Simulation details – The scene is described as a $5\,\mathrm{m} \times 2.5\,\mathrm{m}$ 2D surface. Wall boundary conditions (zero velocity) are applied to the frontiers, except for the top edge, which is an outlet (zero diffusion of flow variables). The propellers are modeled as two squares starting in the middle of the scene, with wall boundary conditions on the left, right, and top edges, and inlet conditions for the bottom edge (normal velocity of intensity proportional to the rotation speed of the propeller). We mesh the scene with triangular cells of an average size of 15 mm, and add inflation near wall boundaries. We let the simulator update the mesh during time with default parameters.

Drone trajectory control – has received special care, and is obtained using model predictive control (MPC) of a dynamical model of a 2D drone allowing realistic trajectory tracking. The model is obtained by constraining the dynamics of a 3D drone model (Romero et al., 2022) to motion in a 2D plane and reducing the number of rotors to two. The drone can therefore move along the axis x and y, and pivot around the z-axis perpendicular to the simulation plane as

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

follows:

$$\begin{cases} \ddot{x} &= -K_1(\Omega_1^2 + \Omega_2^2)\sin(\theta) + K_2(\Omega_1 + \Omega_2)\dot{x} \\ \ddot{y} &= K_1(\Omega_1^2 + \Omega_2^2)\cos(\theta) - g + K_2(\Omega_1 + \Omega_2)\dot{y} \\ \ddot{\theta} &= K_3(\Omega_2^2 - \Omega_1^2), \end{cases} \tag{E-1}$$

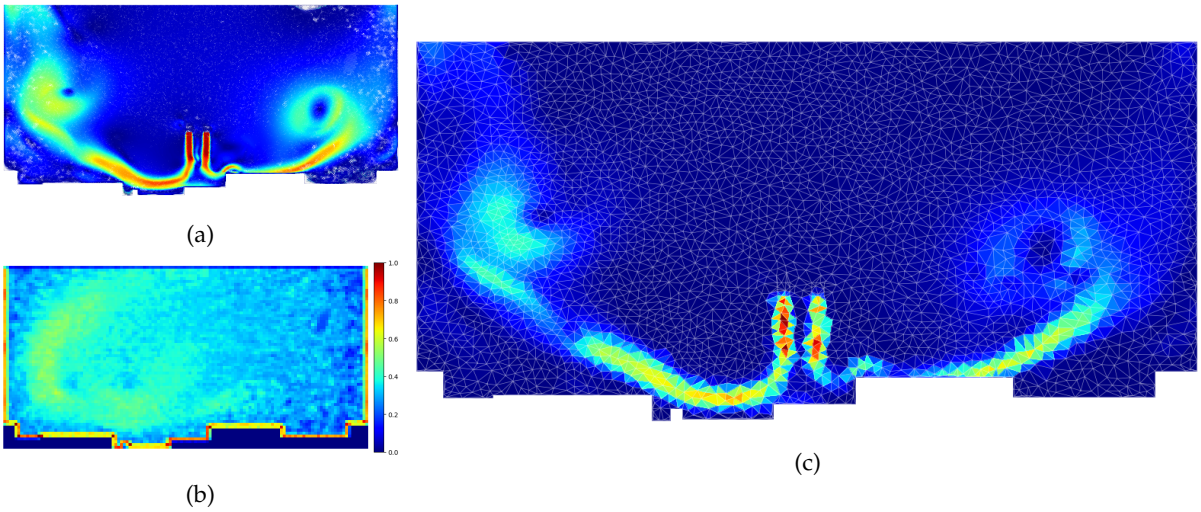
where x, y is the 2D position of the drone and θ its orientation, Ω_1 and Ω_2 the left/right propeller rotation speed, $g = 9.81\text{m/s}$ is acceleration (gravity), and $K_1 = 10^{-4}, K_2 = 5 \times 10^{-5}, K_3 = 5.5 \times 10^{-3}$ are physical constants depending on drone geometry. The resulting trajectories represent physically plausible outcomes, taking into account inertia and gravity.

Mesh down-sampling – consists in simplifying the raw simulation data, as they are not suitable for direct deep learning applications, and require post-processing (see Figure E.1a). The simulation software leverages a very fine-grained mesh dynamically updated to accurately solve the Navier-Stokes equations. The main step thus consists of simplifying the mesh to a reasonable number of nodes. Formally, our goal is to construct a new coarser mesh $(\mathcal{X}^t, \mathcal{E}^t)$ based upon the raw mesh proposed by the simulation software $(\bar{\mathcal{X}}^t, \bar{\mathcal{E}}^t)$. To cope with the dynamic nature of the simulation mesh, our approach consists of dividing the target node set into a static and a dynamic part $\mathcal{X}^t = \mathcal{S} + \mathcal{D}^t$.

- **The static mesh** is obtained by subsampling the simulation point cloud using Poisson Disk Sampling ((Cook, 1986)). However, the spatial density of $\bar{\mathcal{X}}^t$ evolves (certain areas of space are more densely populated at the end of the simulation than at the start). To preserve finer resolution near relevant regions, we thus concatenated 5 regularly spaced point clouds $\bar{\mathcal{X}}^{t_k}$ into a single set. We then sub-sample the resulting set by randomly selecting a point, and deleting all neighbors in a sphere of radius R around the chosen point. This operation is repeated until no more point is at a distance less than R from another. We used an adaptive radius R correlated to the density map: when the original point cloud is dense, the radius is smaller. Conversely, the radius increases in sparse areas. An example of the density map is provided in Figure E.1b.
- **The dynamic mesh** is mandatory to track drone motion accurately. We therefore complete the static mesh with a dynamical part that follows the boundaries of the UAV. To do so, we used the ground truth trajectory to track drone position and orientation across time and extrapolate bounding boxes, which are then transformed into point clouds by sub-dividing the box into several points.

Finally, the edge set $\mathcal{E}^t$ is computed using constrained Delaunay triangulation to prevent triangles from spawning outside of the domain. Once $(\mathcal{X}^t, \mathcal{E}^t$ has been computed, we evaluate the pressure and velocity field $\mathcal{V}^t, \mathcal{P}^t$ on the nodes by averaging the three nearest points in raw simulation data. We illustrate the final result in figure E.1. Better mesh simplification algorithms exist, notably minimizing the interpolation error, yet such algorithms rely on the simulated flow to compute the mesh, which may embed unwanted biases or shortcuts in the mesh geometry.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) (b) (c)

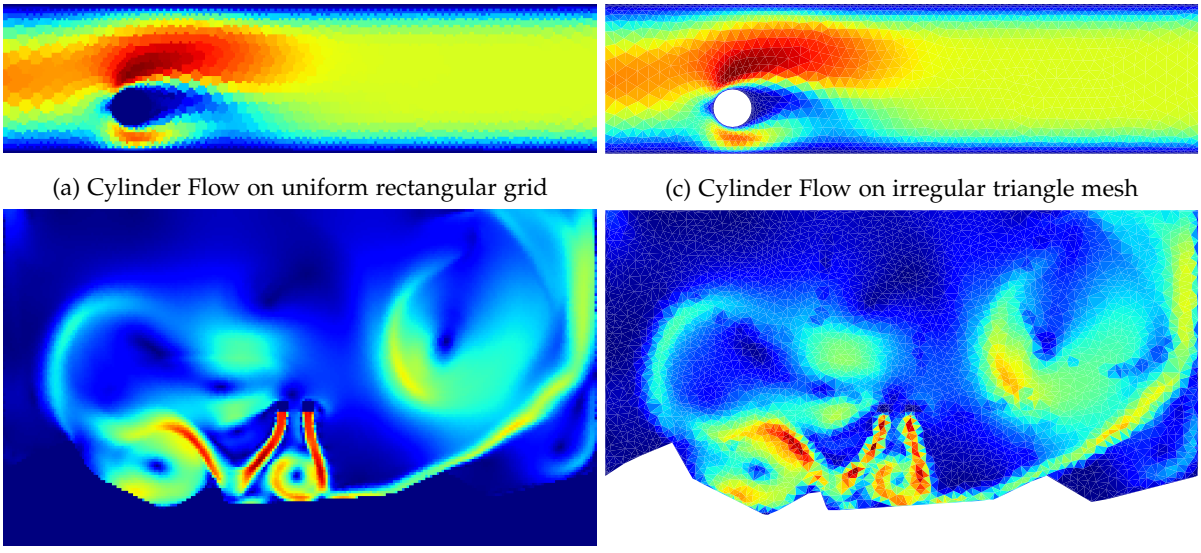
Figure E.1: **Down-sampling from raw simulation** – (a) sample of raw simulation measurements obtained on a high-resolution mesh. This single snapshot contains 158,961 nodes. (b) example of a node density map controlling the sampling disk radius. The raw mesh is dense near the boundaries and on the left side, as this sample is taken from a simulation where the drone explores the left region of the scene. (c) Mesh simulation at final resolution. We drastically simplified the mesh while maintaining a satisfactory level of detail.

E.1.2 *Grid based dataset*

One of the baselines, DilResNet (Stachenfeld et al., 2021), relies on convolutional layers for future forecasting of turbulent flows and therefore requires projecting *Eagle* and Cylinder-Flow on a uniform rectangular grid. However, such a discretization scheme can not adapt its spatial resolution as a function of the geometry of the scene, which therefore constitutes a disadvantage with respect to an irregular triangular mesh. To limit this effect, the resolution of the grid is chosen such that the number of pixels is at least ten times larger than the number of points in the triangular mesh.

We project Cylinder-Flow onto a uniform 256×64 grid and *Eagle* onto a 256×128 grid (the dimensions were chosen to respect the height-width ratio of the original data). The value of the pressure and velocity fields at each point in the grid is extrapolated from the nearest point in the raw simulated data. We illustrate this projection in figure E.2. While the grid-based simulation (figure E.2b) seems visually more accurate than the mesh-based simulation (figure E.2d), we observed that the re-projection error (i.e. the error obtained after projecting the grid-based data onto the triangular mesh) is greater near sensible regions, for example near the scene boundaries.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés



(a) Cylinder Flow on uniform rectangular grid

(c) Cylinder Flow on irregular triangle mesh

(b) *Eagle* on uniform rectangular grid

(d) *Eagle* on irregular triangle mesh

Figure E.2: **Mesh to grid conversion** – Illustration of the pixellisation process. The left column (a and b) shows snapshots of simulations from the grid-based datasets, used to train DilResNet. For comparison, we show the same snapshots in the mesh-based datasets (c and d). While resolution seems better on grid-based simulation, it lacks precision near sensible regions, which are primordial for accurate forecasts.

E.2 Model details

E.2.1 *Clustering*

We use our own implementation of the *same size Kmeans* algorithm described here[1]. Using equally sized clusters has two main advantages :

- Areas of high density will be covered by a greater number of clusters, allowing the adaptive resolution of irregular meshes to be preserved on the coarser mesh.
- The model can be implemented efficiently, maximizing parallelization, since clusters can be easily stored as batched tensors.

Since the clustering depends solely on the geometric properties of the mesh (and not on the prediction of the neural network), it is possible to apply the clustering algorithm as a pre-processing step to reduce the computational burden during training. Note that since the mesh is dynamic, so are the clusters: the k^e cluster at time t will not necessarily contain the same points at time $t+1$.

E.2.2 *Architecture and Training Details*

We kept the same training setup for all datasets and trained our model for 10,000 steps with the Adam optimizer and a learning rate of 10^{-4} to minimize (8-6) with $\alpha = 10^{-1}$ and $H = 8$.

[1] https://elki-project.github.io/tutorial/same-size_k_means

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Velocity and pressure are normalized with statistics computed on the train set, except for Scalar-Flow, where better results are obtained without normalization.

Encoder – ϕ_{node} and ϕ_{edge} are one-layer MLPs with ReLU activations, hidden size and output size of 128 ($(\eta_i, e_{ij}) \in \mathbb{R}^{128}$). We used L=4 chained graph neural network layers composed of two identical MLPs ψ_{edge} and ψ_{node} with two hidden layers of dimension 128, ReLU activated, followed by layer normalization. The positional encoding function F is defined as follows:

$$F(\boldsymbol{x}) = [\cos(2^i \pi \boldsymbol{x}) \ \sin(2^i \pi \boldsymbol{x})]_{i=-3,\dots 3} \tag{E-2}$$

where $\boldsymbol{x}$ is a 2D vector modeling the position of node i.

Graph Pooling – we used a single layer gated recurrent unit with a hidden size of dimension W followed by a single layer MLP with hidden and output size of W. This step produces a cluster feature representation $\boldsymbol{w}_k \in \mathbb{R}^W$. For Cylinder-Flow and *Eagle*, W=512. For Scalar-Flow, W=128.

Attentional module – following (Xiong et al., 2020) an attention block is defined as follows for an input $w \in \mathbb{R}^W$:

$$\begin{aligned}
\boldsymbol{w}_1 &= [\text{LN}(\boldsymbol{w}) \ F(\bar{\boldsymbol{x}}_k)] \\
\boldsymbol{w}_2 &= \text{MHA}(\boldsymbol{w}_1, \boldsymbol{w}_1, \boldsymbol{w}_1) \\
\boldsymbol{w}_3 &= \boldsymbol{w} + \text{Linear}(\boldsymbol{w}_2) \\
\boldsymbol{w}_4 &= \text{LN}(\boldsymbol{w}_3) \\
\boldsymbol{w}_5 &= \text{MLP}(\boldsymbol{w}_4) \\
\boldsymbol{w}_6 &= \boldsymbol{w}_3 + \boldsymbol{w}_5
\end{aligned} \tag{E-3}$$

where LN are layer norm functions, Linear is a linear function (with bias), MHA is multi-head attention and MLP is a multi-layer perceptron with one hidden layer of size W. We denote the barycenter of cluster k as $\bar{\boldsymbol{x}}_k$. We used M=4 chained attention block, with four attention heads each. The last attention layer is followed by a final layer norm.

Decoder – the decoder takes as input the node embeddings η_i, the cluster features updated by the attentional module $\boldsymbol{w}_k^M$ and the node-wise positional encoding $\boldsymbol{f}_i$. We applied a graph neural network composed of two identical MLPs (two hidden layers with the hidden size of 128, ReLU activated, and layer norm). The resulting node embeddings are fed to a final MLP with two hidden layers and a hidden size of 128, with TanH as the activation function.

E.2.3 *Baselines training details*

After performing a grid search to select the best options, we found that training each baseline to minimize (8-6) with Adam optimizer and learning rate of 10^{-4} produces the best results. We vary the weighting factor α to maintain the balance between pressure and velocity. For Cylinder-Flow and *Eagle*, we trained the baselines over $H = 5$ time-steps. For Scalar-Flow, we set $H = 20$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

- **MeshGraphNet:** we performed a grid search over the number of GNN layers to fit each dataset, but the best results were obtained with the recommended depth $L = 15$ for each dataset. Conversely to what is suggested in Pfaff et al. (2020), we found that training MeshGraphNet over a longer horizon improves the general performances. We used our own implementation of the baseline and made sure to reproduce the results presented in the main paper (for Cylinder-Flow only). We get the best trade-off between velocity and pressure with $\alpha = 10$.
- **GAT:** we performed a grid search over the number of heads per layer and the number of layers. The best results were obtained for 10 layers of graph attention transformer and two attention heads per layer (except for Cylinder-Flow, where four heads slightly improve the performances).
- **DilResNet:** we found that increasing the number of blocks improves overall performance, setting the number of convolutional blocks from 4 to 20.

The baselines are structurally built to predict velocity field $\mathcal{V}_t^{t+h}$ and pressure field $\mathcal{P}_t^{t+h}$ described on the mesh geometry at current time $\mathcal{X}^t$. Auto-regressive forecasting on a longer horizon thus requires interpolation of the predicted flow to the (provided) future mesh $\mathcal{X}^{t+h}$. We do not want interpolation to disturb our problem of interest, which is turbulent flow prediction. Therefore, we made the interpolation from $\mathcal{X}^t$ to time $\mathcal{X}^{t+1}$ straightforward. As the vast majority of the mesh remains static (see previous section), only the nodes linked to the UAV need to be interpolated. Since they can readily be associated in a one-to-one relation, nearest point interpolation can be performed automatically by assigning $\mathcal{V}_t^{t+h}$ at these points to $\mathcal{V}^{t+h}$.

E.3 More results

E.3.1 *Detailed metrics*

Formally, we used the following metrics to report our results on the test set $\mathcal{D}$:

$$\text{N-RMSE} = \frac{1}{H|\mathcal{D}|} \sum_{\mathcal{D}} \sum_{t=1}^{H} \frac{|\boldsymbol{v}^t - \hat{\boldsymbol{v}}^t|}{\tilde{\boldsymbol{v}}} + \frac{|p^t - \hat{p}^t|}{\tilde{p}} \tag{E-4}$$

where $\tilde{\boldsymbol{v}}$ and $\tilde{p}$ are the standard deviation of velocity and pressure field computed on the train set.

Detailed metric – the raw root mean squared error (RMSE) on each field is reported in figure E.4 as well as the temporal evolution of N-RMSE across prediction horizon. On Cylinder-Flow (Figure E.4a), velocity error is very similar between MeshGraphNet and ours. Our model slightly outperforms the baseline on the pressure field, yielding overall better performances. However, the temporal evolution of the N-RMSE indicated that both models converge to the same accuracy for very long roll-out prediction. On *Eagle*, our model shows excellent stability over a long horizon and produces accurate velocity and pressure estimates.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

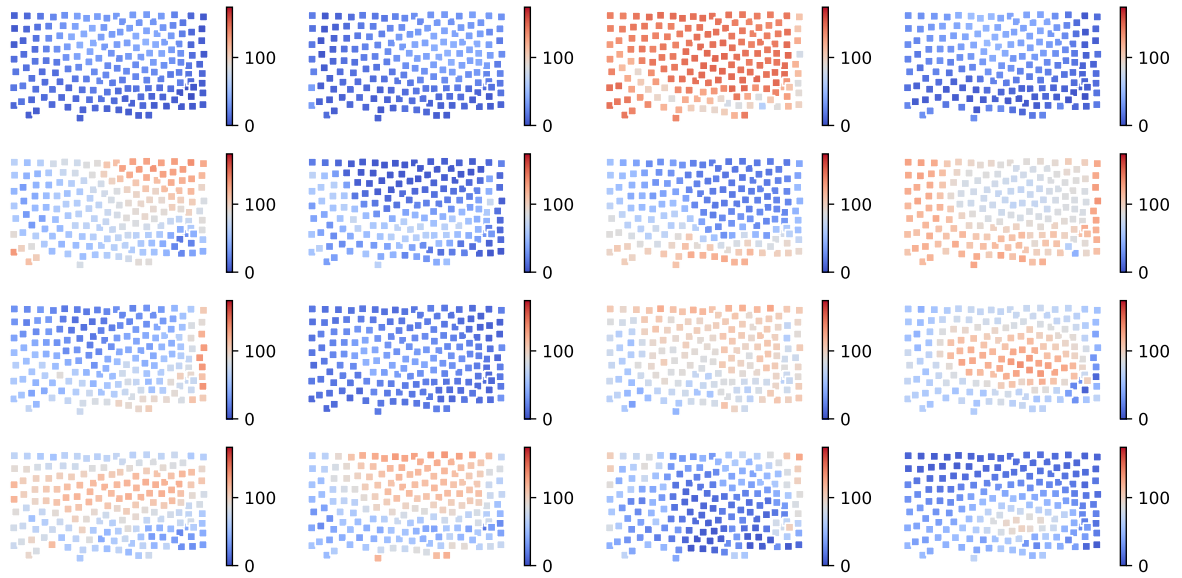
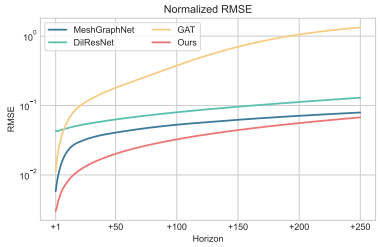


Figure E.3: **K-number maps** computed for each cluster, that is, the number of nodes required to reach 90% of the attention. A low k-number indicates a very specialized head (attending to a few nodes), while a high k-number indicates uniform attention.

K-number – is a property which can be calculated for attention maps, and which consists in the number of tokens required to reach 90% of attention (Kervadec et al., 2021). This property can be used to characterize the shape of attention maps, varying from peaky attention (requiring few tokens to reach 90%) to more uniform attention heads. We show k-numbers in Figure E.3. Interestingly, the k-number maps can be compared with attention maps figure 8.5d: peaky heads (in blue) are correlated with relatively local attention maps, and conversely, more uniform heads (in red) correspond to attention maps focusing on larger distances, often following the airflow. Some heads have different behaviors depending on the selected cluster and are peaky in some areas (mainly around the boundaries of the scene), but more uniform elsewhere. These cues support the importance of global attention in our model.

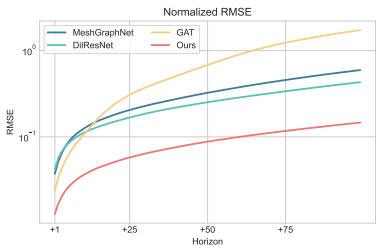
Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Horizon	+1		+50		+250	
Field	V	P	V	P	V	P
MGN	0.0004	0.0016	0.0047	0.0095	**0.0144**	0.0145
GAT	0.0015	0.0025	0.0278	0.0360	0.2595	0.2314
DRN	0.0098	0.0063	0.0152	0.0085	0.0344	0.0152
Ours	**0.0003**	**0.0007**	**0.0044**	**0.0035**	0.0179	**0.0079**



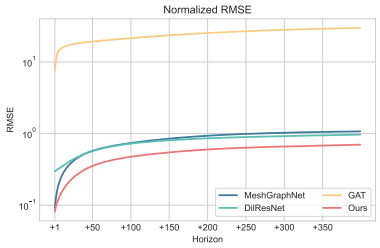
(a) **CylinderFlow**: (Right) RMSE on velocity **V** and pressure **P** fields. (Left) Normalized RMSE over the forecasting horizon. Our mesh transformer overcomes the baselines by a small margin. Yet qualitative results tend to indicate that Cylinder Flow is already a well-mastered task.

Horizon	+1		+50		+100	
Field	V	D	V	D	V	D
MGN	0.0009	0.0059	0.0105	0.0568	0.0231	0.1130
GAT	0.0009	0.0066	0.0097	0.0578	0.0200	0.1091
DRN	0.0014	0.0101	0.0130	0.0750	0.0237	0.1217
Ours	**0.0005**	**0.0024**	**0.0035**	**0.0145**	**0.0059**	**0.0239**



(b) **Scalar Flow**: (Right) RMSE on velocity **V** and density **D** fields. (Left) Normalized RMSE over the forecasting horizon. Our model shows improvements over the baselines in both fields.

Horizon	+1		+50		+250	
Field	V	P	V	P	V	P
MGN	0.0810	**0.4256**	0.5926	2.2492	1.0702	3.7220
GAT	0.1698	64.546	0.8551	162.56	1.0959	227.20
DRN	0.2517	1.4453	0.5374	2.4568	0.9188	3.5824
Ours	**0.0537**	0.4590	**0.3494**	**1.4432**	**0.6826**	**2.4130**



(c) ***Eagle***: (Right) RMSE on velocity **V** and pressure **P** fields. (Left) Normalized RMSE over the forecasting horizon. Our largely and consistently and reliably outperforms the competing baselines. While MeshGraphNet and DilResNet show comparable performances during the first time steps, our model succeeded in controlling error accumulation for reasonable horizons and eventually presented better simulations.

Figure E.4: **Detailed metrics** – on Cylinder-Flow, Scalar-Flow and *Eagle*, evaluated for each baselines and our model.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

E.3.2 *Qualitative results*

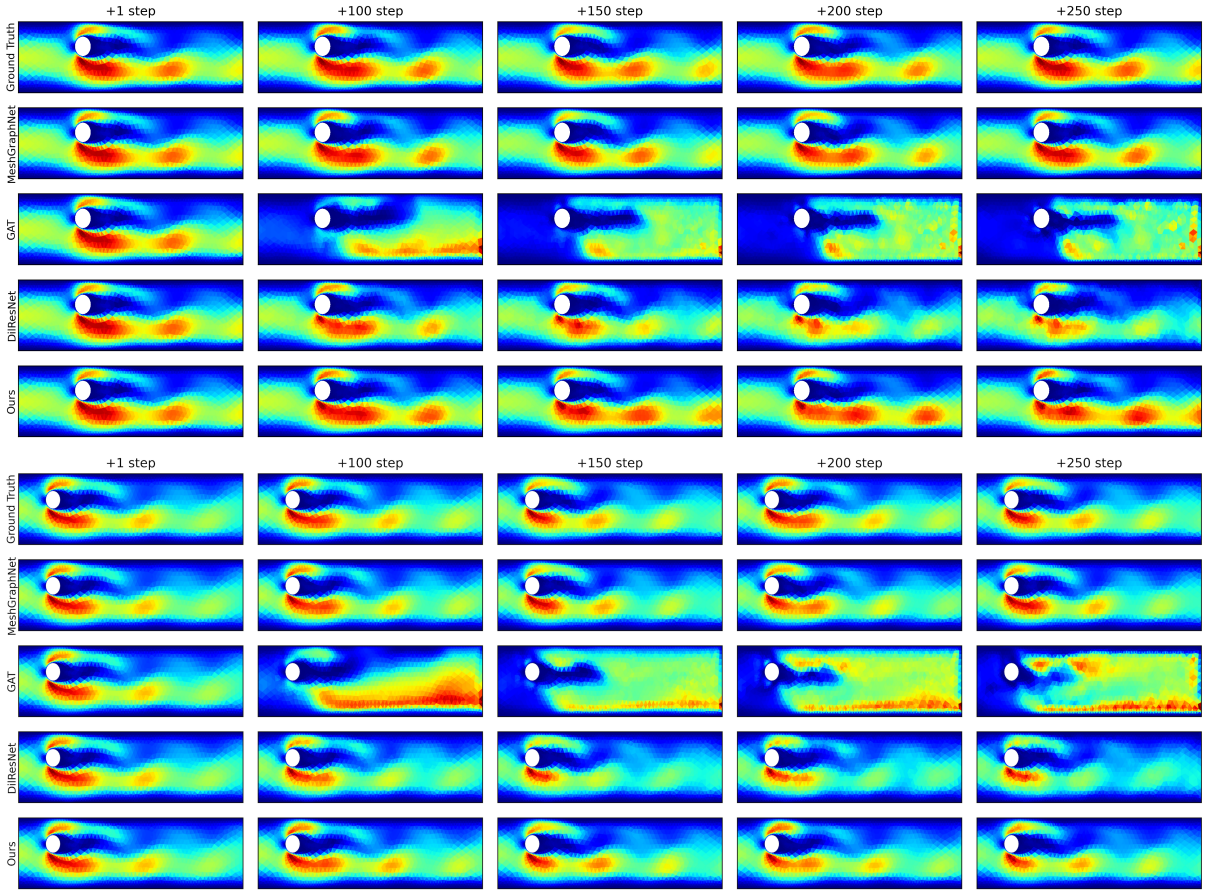


Figure E.5: Examples of prediction forward in time on Cylinder-Flow

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

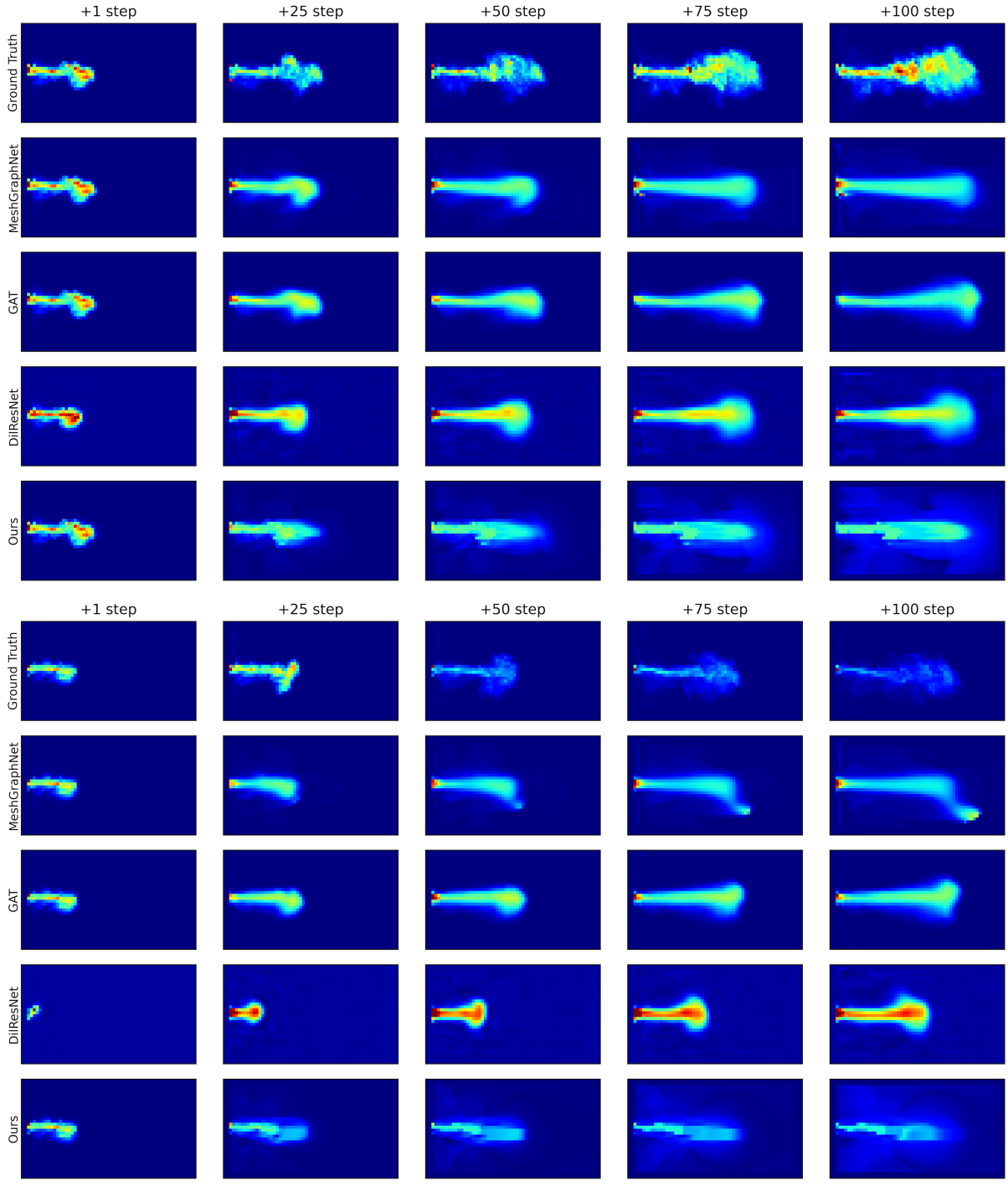


Figure E.6: Examples of prediction forward in time on Scalar-Flow

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

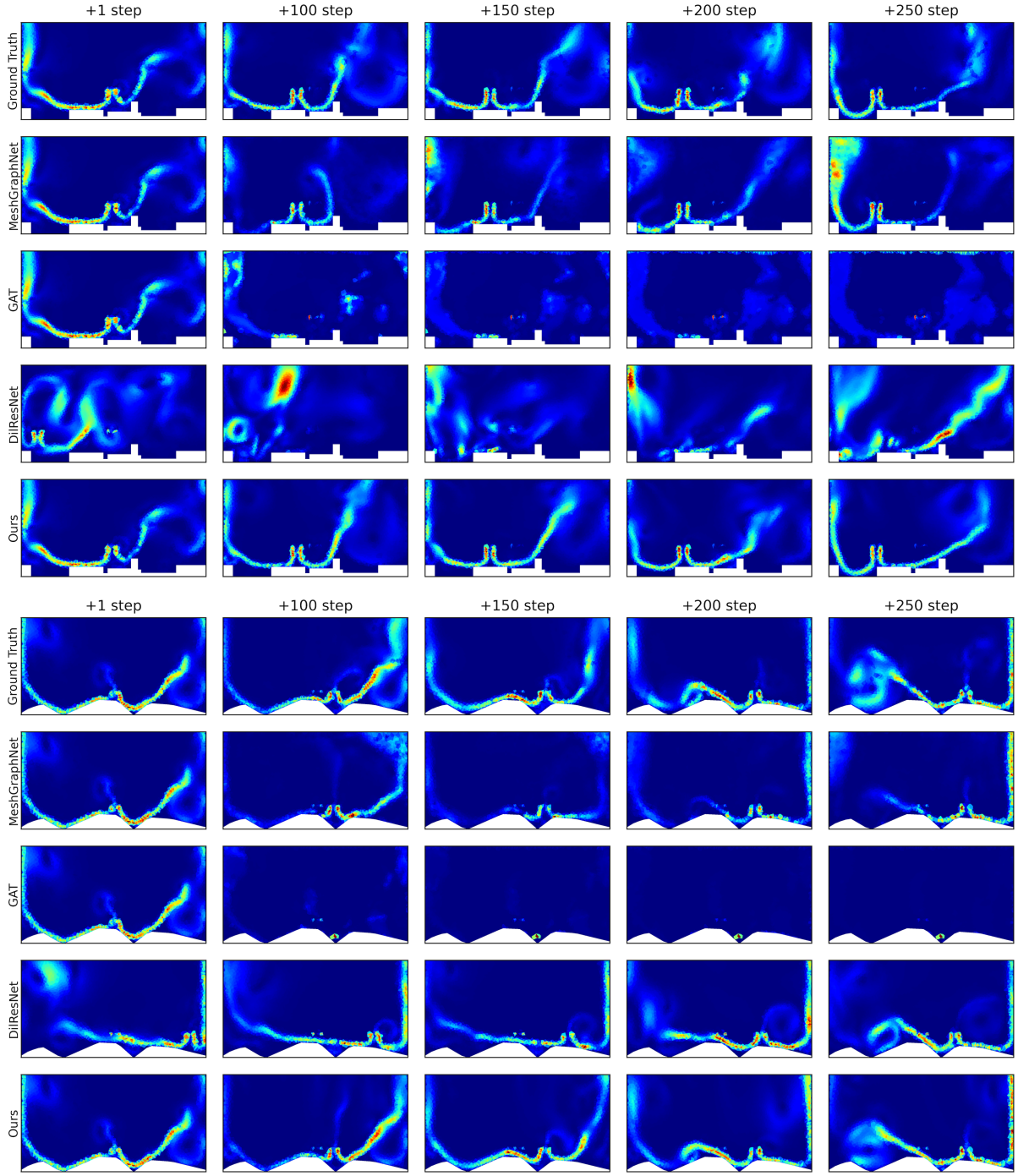


Figure E.7: Examples of prediction forward in time on *Eagle*

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

E.3.3 *Failure case*

Despite the excellent performance of our model against competitive baselines, there is still room for improvement. Some more difficult configurations give rise to very turbulent flows, widely extended in the scene. The evolution of these flows is more difficult to predict and the models we evaluated failed to remain accurate. In these cases, the precision with which the small vortices are simulated is essential, because some of them will grow to become the majority.

Moreover, our model suffers from an error accumulation problem, like any auto-regressive model. Experimentally, we observe that the airflow tends to be smoothed by deep learning models when the prediction horizon increases.

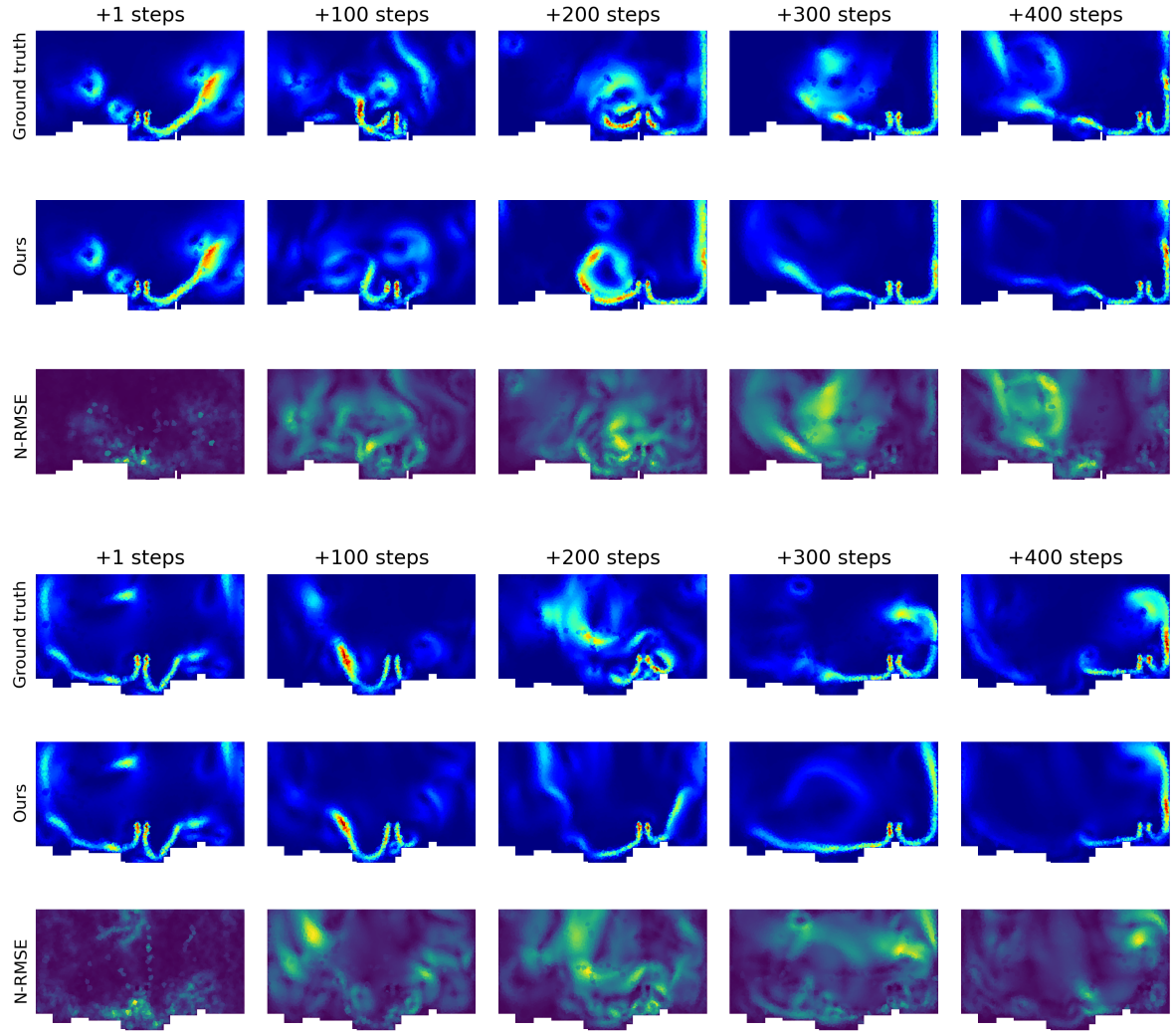


Figure E.8: **Failure cases** – We expose failure cases of our mesh transformer on Eagle. The error increases when the flow tends to intensify throughout the scene, and when turbulence dominates. Over a longer prediction horizon, the airflow tends to be smoother and less turbulent.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

N-RMSE		Training			
		90%	80%	70%	60%
Testing	90%	**0.454**	0.497	0.513	0.502
	80%	**0.427**	0.446	0.467	0.440
	70%	0.406	0.405	0.416	**0.394**
	60%	0.401	0.370	0.368	**0.348**

Table E.1: **Mesh down-sampling** – We train our mesh transformer under different regimes of down-sampling by keeping a fixed percentage of points from the initial mesh and removing the others. We evaluate the resulting models on regimes different from training and observe very little variations in N-RMSE among them.

E.3.4 *Generalization to different mesh resolution*

In *Eagle*, the number of points varies from one simulation to another, forcing the model to generalize on meshes of different sizes. We explicitly demonstrate the performance of our mesh transformer on this task in table E.1. Four instances of the model are trained on a particular regime in which the simulation meshes are randomly down-sampled, respectively at 90%, 80%, 70%, and 60% of the initial mesh resolution. These models are then evaluated in a different regime from the one used for training, either higher (more points on average during the test than during training) or lower (fewer points on the test than in training). We show that our model generalizes well to these different regimes by giving relatively close N-RMSE measurements for a given down-sampling regime.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Appendix F

Appendix for chapter 9

F.1 Further details on dataset generation

Confounders – in our setup are masses, which we discretize in $\{1, 10\}$. For `BallsCF` and `CollisionCF`, we can also consider the continuous initial velocities of each object as confounders variables, since they have to be identified in **AB** to forecast CD. We simulate all trajectories associated with the various possible combinations of masses from the same initial condition.

Do-interventions – however, depend on the task. For `BlocktowerCF` and `BallsCF`, do-interventions consist of (a) removing the top cube or a ball, or (b) shifting a cube/ball on the horizontal plane. In this case, for `BlocktowerCF`, we make sure that the cube does not move too far from the tower, to maintain contact. For `CollisionCF`, the do-interventions are restricted to shifting operations, since there are only two objects (a ball and a cylinder). It can consist of either a switch of the orientation of the cylinder between vertical or horizontal, or a shift of the position of the moving object relative to the resting one in one of the three canonical directions x, y, and z.

F.1.1 *Enforcing the Identifiabilty constraint*

The identifiability and counterfactuality constraints described in section 9.2 are imposed numerically, i.e. we first sample and simulate trajectories with random parameters and then reject those that violate these constraints.

As stated in section 9.2, an identifiable experiment guarantees that there is no pair (z, z') that gives the same trajectory **AB** but a different counterfactual outcome CD. Otherwise, there will be no way to choose between z and z' only from looking at **AB**, thus no way to correctly forecast the counterfactual experiment. By enforcing this constraint, we make sure that there exists at least a set $\{z, z', ...\}$ of confounders that give at the same time similar observed outcomes **AB** and similar counterfactual outcomes CD.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

In practice, there exists a finite set of possible variables z_i, corresponding to every combination of masses for each object in the scene (masses take their value in $\{1, 10\}$). During generation, we submit each candidate experiment (**AB**, CD, z) to a test ensuring that the candidate is identifiable. Let $\psi(X_0, z)$ be the function that gives the trajectory of a system with initial condition X_0 and confounders z. We simulate all possible trajectories $\psi(A, z_i)$ and $\psi(C, z_i)$ for every possible z_i. If there exists $z' \neq z$ such that the experiment is not identifiable, the candidate is rejected. This constraint requires simulating the trajectory of each experiment several times by modifying the physical properties of the objects.

Equalities in Definition 9.1 are relaxed by thresholding distances between trajectories. We reject a candidate experiment if there exists a z' such that

$$\sum_{t=0}^{T} \left|\psi(A, z) - \psi(A, z')\right| < \varepsilon \text{ and } \sum_{t=0}^{T} \left|\psi(C, z) - \psi(C, z')\right| > \varepsilon. \tag{F-1}$$

The choice of the threshold value ε is critical, in particular for the identifiability constraint:

- If the threshold is **too high**, all **AB** trajectories will be considered equal, which results in the acceptance of unidentifiable experiments.
- If the threshold is **too low**, all trajectories CD are considered equal. Again, this leads to mistakenly accepting unidentifiable experiments.

There exists an optimal value for ε, which allows rejecting unidentifiable experiences. To measure this optimal threshold, we generated a small instance of the `BlocktowerCF` dataset without constraining the experiments, i.e. trajectories can be unidentifiable and non-counterfactual. We then plot the percentage of rejected experiments in this unfiltered dataset against the threshold value (Fig. F.1, left). We chose the threshold $\varepsilon = 100$ which optimizes discrimination and rejects the highest number of "unidentifiable" trajectories.

To demonstrate the importance of this, we train a recurrent GNN on `BlocktowerCF` to predict the cube masses from ground-truth state trajectories **AB**, including pose and velocities, see Fig. F.2. It predicts the mass of each cube by solving a binary classification task. We train this model on both `BlocktowerCF` and an alternative version of the scenario generated without the identifiability constraint. The results are shown in Table F.1a. We are not aiming for 100% accuracy, and this problem remains difficult in the sense that the identifiability constraint ensures the identifiability of a set of confounder variables, while our sanity check tries to predict a unique z.

However, the addition of the identifiability constraint to the benchmark significantly improves the model's accuracy, which indicates that the property acts positively on the feasibility of Filtered-CoPhy. The *corrected accuracy* metric focuses solely on the critical cubes, i.e. those cubes whose masses directly define the trajectory CD.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

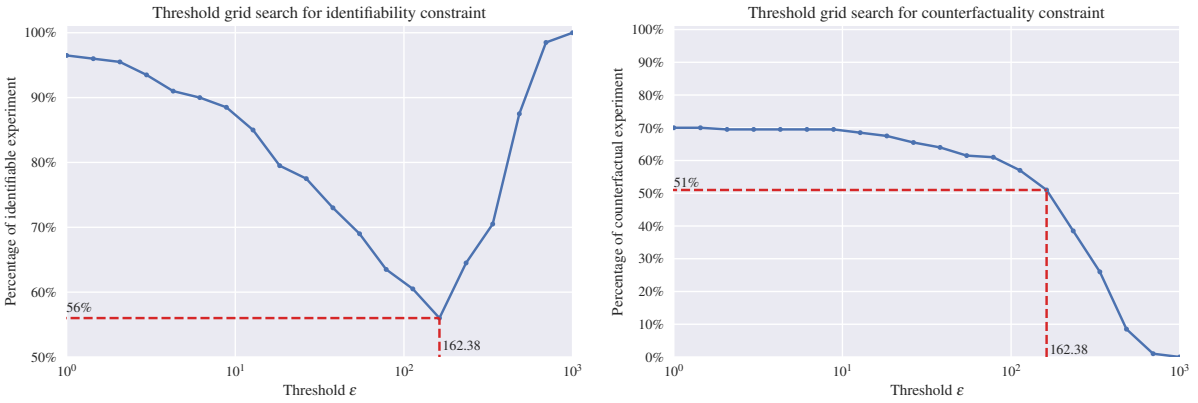


Figure F.1: **Experimental tuning of the threshold parameter** – We generate an unconstrained subset of `BlocktowerCF` and plot the percentage of identifiable experiments as a function of the threshold ε.

	without constraint	with constraint
Accuracy	56%	**84%**
Corrected Acc.	58%	**91%**

(a)

FPS	5	15	25	35	45
MSE ($\times 10^{-2}$)	4.58	3.97	**3.74**	3.82	3.93

(b)

Table F.1: **Sanity check** – (a) identifiability constraint in `BlocktowerCF`, which results in better estimation of cube masses. The corrected accuracy only considers those cubes for which changes in masses are consequential for the trajectory **D**. (b) MSE between ground truth 3D positions and predicted positions after 1 second, depending on the sampling rate of the trajectory.

F.1.2 *Enforcing the Counterfactuality constraint*

Let $(\mathbf{AB}, \mathbf{CD}, z)$ be a candidate experiment, and z^k be a combination of masses identical to z except for the kth value. The counterfactuality constraint consists of checking that there exists at least one k such that $\psi(C, z) \neq \psi(C, z^k)$. To do so, we simulate $\psi(C, z^k)$ for all k and measure the difference with the candidate trajectory $\psi(C, z)$. Formally, we verify the existence of k such that:

$$\sum_{t=0}^{T} \left| \psi(C, z^k) - \psi(C, z) \right| < \varepsilon. \tag{F-2}$$

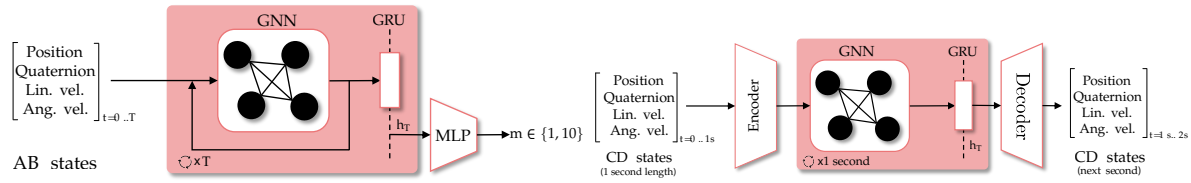


Figure F.2: **Sanity check** – Impact of the choice of temporal resolution. Left: We check the identifiability constraint by training a model to predict the cube masses in `BlocktowerCF` from the observed trajectory **AB**. The model is a graph neural network followed by a gated recurrent unit. Right: We check the augmentation of the sampling rate by training an agent to forecast a 1-second-length trajectory from the states of the previous second.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

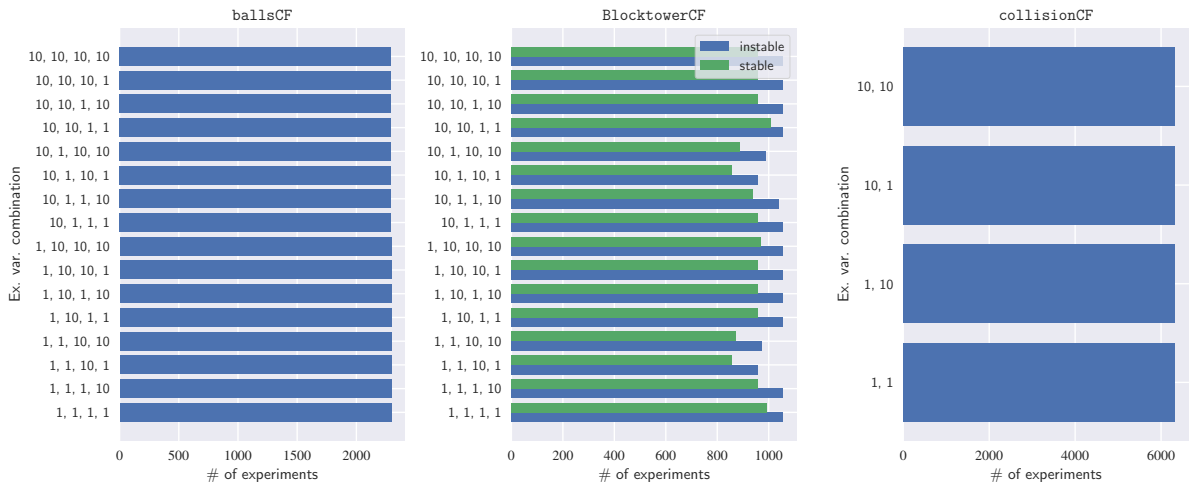


Figure F.3: **Dataset balance** – During the dataset generation process, we carefully balance combinations of masses, i.e. the confounders. For `BlocktowerCF`, we also guarantee that the proportion of stable CD towers is close to 50% for each confounder configuration.

F.1.3 *Analyzing temporal resolution*

We analyzed the choice of temporal frequency for the benchmark with another sanity check. We simulate a non-counterfactual dataset from `BlocktowerCF` where all cubes have equal masses. A recurrent graph network takes as input cube trajectories (poses and velocities) over a time interval of one second and predicts the rest of the trajectory over the following second. We vary the sampling frequency; for example, at 5 FPS, the model receives 5 measurements and predicts the next 5 time steps, which correspond to a one-second rollout in the future. Finally, we compare the error in 3D positions between the predictions and the ground truth on the last prediction. Results are shown in Table F.1b. This check shows clearly that 25 FPS corresponds to the best trade-off between an accurate representation of the collision and the amount of training data.

F.1.4 *Simulation details*

We used Pybullet as a physics engine to simulate Filtered-CoPhy. Each experiment is designed to respect the balance between good coverage of confounder combinations and counterfactuality and identifiability constraints described above. We generate the trajectories iteratively:

1. We sample a combination of masses and other physical characteristics of the given experiment, such as stability of the tower, object motion in `CollisionCF`, or if the do-operation consists of removing an object. This allows us to maintain a balance of confounder configurations.
2. Then we search for an initial configuration **A**. For `BlocktowerCF`, we make sure that this configuration is unstable to ensure identifiability. Then we simulate the trajectory **B**.
3. We look for a valid do-operation such that identifiability and counterfactuality con-

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

# Keypoints		K	2K	4K
BT-CF	PSNR	34.40	**35.41**	34.92
	MSE Grad	27.24	**21.39**	23.99
B-CF	PSNR	**37.76**	37.06	36.98
	MSE Grad	**3.47**	3.77	3.95
C-CF	PSNR	32.00	**35.41**	34.42
	MSE Grad	32.00	**12.57**	17.09

Table F.2: **Reconstruction task** – PSNR (dB) on the task of reconstructing the target from the source (both randomly sampled from Filtered-CoPhy), using 5 coefficients per keypoint. We vary the number of keypoints in our model. Here K is the maximum number of the objects in the scene.

straints are satisfied. If no valid do-operation is found after a fixed number of trials, we reject this experiment.

4. If a valid pair (**AB**, **CD**) is found, we add the sample to the dataset.

The trajectories were simulated with a sample time of 0.04 seconds. The video resolution is 448×448 and represents 6 seconds for `BlocktowerCF` and `BallsCF`, and 3 seconds for `CollisionCF`. We invite interested readers to look at our code for more details, such as do-operation sampling, or intrinsic camera parameters. Fig. F.3 shows the confounder distribution in the three tasks.

F.2 Performance evaluation of the de-rendering module

F.2.1 *Image reconstruction*

We evaluate the reconstruction performance of the de-rendering module in the reconstruction task. Note that there is a trade-off between the reconstruction performance and the dynamic forecasting accuracy: a higher number of keypoints may lead to better reconstruction, but can hurt prediction performance, as the dynamic model is more difficult to learn.

Reconstruction error – we first investigate the impact of the number of keypoints in Table F.2 by measuring the Peak Signal to Noise Ratio (PSNR) between the target image and its reconstruction. We vary the number of keypoints among multiples of K, the maximum number of objects in the scene. Increasing the number of keypoints increases reconstruction quality (PSNR) up to a certain point, but results in degradation in forecasting performance. Furthermore, doubling the number of keypoints only slightly improves reconstruction accuracy. This tends to indicate that our additional coefficients are already sufficient to model finer-grained visual details. Table 9.3 in the main paper measures the impact of the number of keypoints and the presence of the additional appearance coefficients on the full pipeline including the dynamic model. Table F.3 illustrates the impact of the number of keypoints and the additional appearance coefficient on the reconstruction performance alone. As we can see, the addition of the coefficient consistently improves PSNR for low numbers of keypoints (over 2 dB for K

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

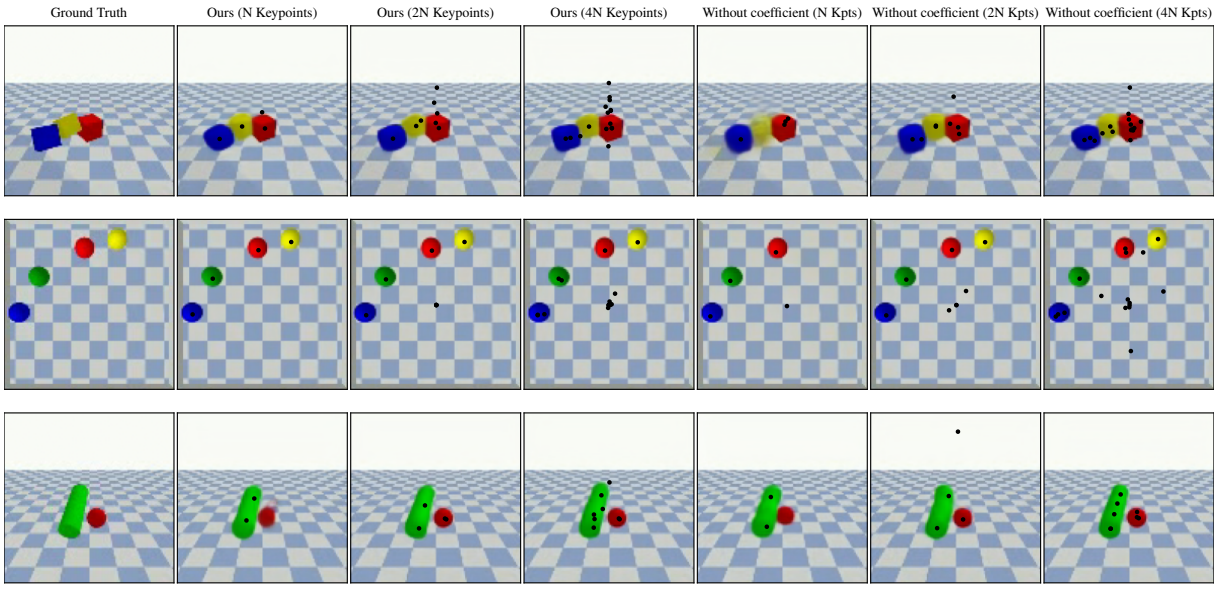


Figure F.4: **Reconstructions** – produced by the de-rendering module. Our model correctly marks each object in the scene and achieves satisfactory reconstruction.

keypoints). The improvement is less visible for larger numbers of keypoints since 3D visual details could be encoded via the keypoint's position, hence coefficients become less relevant. Visualizations are shown in Fig. F.4.

		K		2K		4K	
Coefficients		✗	✓	✗	✓	✗	✓
BT-CF	PSNR	32.53	**34.40**	33.97	**35.41**	34.57	**34.92**
	MSE Grad	41.86	**27.24**	35.24	**21.39**	28.06	**23.99**
B-CF	PSNR	34.62	**37.76**	36.94	**37.06**	**37.15**	36.98
	MSE Grad	6.22	**3.47**	4.16	**3.77**	**4.07**	3.95
C-CF	PSNR	30.65	**32.00**	33.89	**35.41**	**35.63**	34.42
	MSE Grad	12.78	**32.00**	20.59	**12.57**	**11.72**	17.09

Table F.3: **Number of coefficients** – Impact of the number of keypoints and the presence of additional appearance coefficient in the de-rendering module for pure image reconstruction (no dynamic model). We report PSNR (dB) and MSE on the image gradient. K is the maximum number of the objects in the scene. The coefficients significantly improve the reconstruction on a low number of keypoints. This table is related to table 9.3 in the main paper, which measures this impact on the full pipeline.

F.2.2 *Navigating the latent coefficient manifold*

We evaluate the influence of the additional appearance coefficients on our de-rendering model by navigating its manifold. To do so, we sample a random pair $(X_{\text{source}}, X_{\text{target}})$ from an experiment in `BlocktowerCF` and compute the corresponding source features and target keypoints and coefficients. Then, we vary each component of the target keypoints and coefficients and observe the reconstructed image (fig. F.5). We observed that the keypoints accurately control the position of the cube along both spatial axes. The rendering module does infer some

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

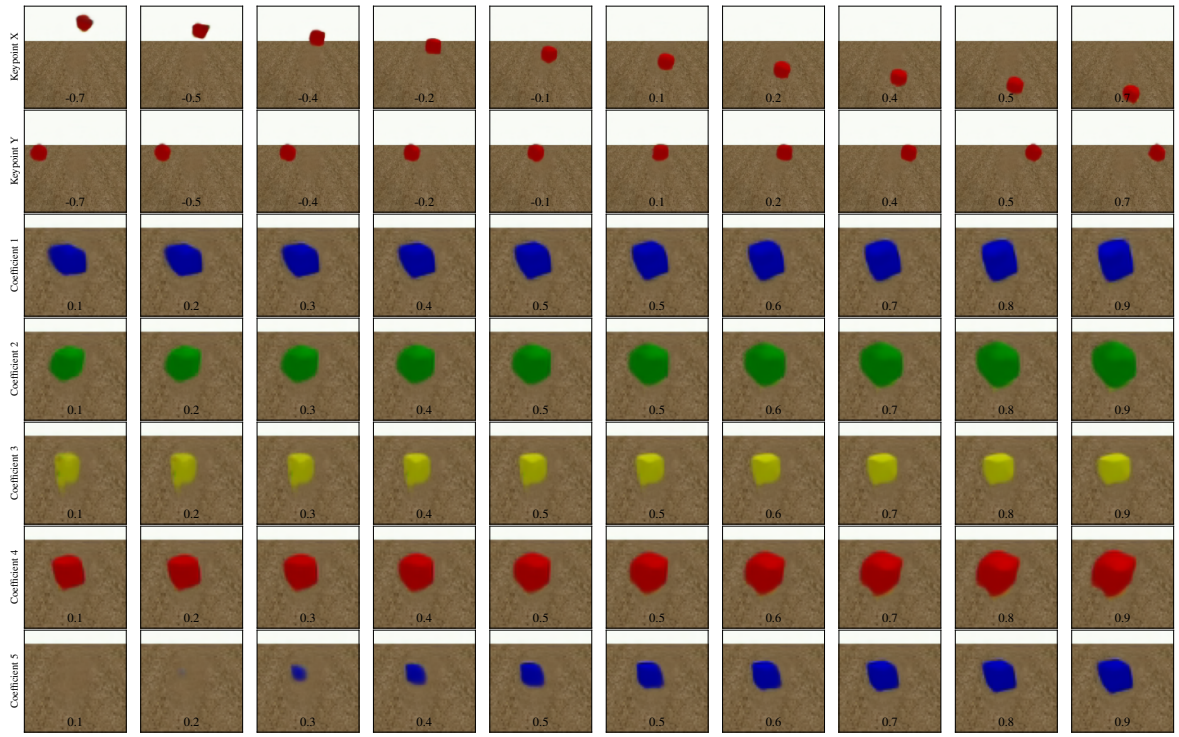


Figure F.5: **Coefficient sweep** – Navigating the manifold of the latent coefficient representation. Each line corresponds to variations of one keypoint coordinate or coefficient and shows the effect on a single cube.

hints on 3D shape information from the vertical position of the cube, exploiting a shortcut in learning. On the other hand, while not being supervised, the coefficients naturally learn to encode different orientations in space and distance from the camera. Interestingly, a form of disentanglement emerges. For example, coefficients n° 1 and 2 control rotation around the z-axis and coefficient $n°4$ models rotation around the y-axis. The last coefficient represents both the size of the cube and its presence in the image.

F.3 Comparison with the Transporter baseline

F.3.1 *Comparison with our de-rendering model*

As described in section 9.3.1, the Transporter (Kulkarni et al., 2019) is a keypoint detection model somewhat close to our de-rendering module. It leverages the transport equation to compute a reconstruction vector:

$$\hat{\Psi}_{\text{target}} = \boldsymbol{f}_{\text{source}} \times (1 - K_{\text{source}}) \times (1 - K_{\text{target}}) + \boldsymbol{f}_{\text{target}} \times K_{\text{target}}. \tag{F-3}$$

where $K_{\text{source}} = \sum_n \mathcal{G}(\boldsymbol{k}_n^{\text{source}})$. This equation allows transmitting information from the input by two means: the 2D position of the keypoints (K_{target}) and the dense visual features of the target ($\boldsymbol{f}_{\text{target}}$). In comparison, our de-rendering solely relies on the keypoints from the target image and does not require a dense vector to be computed on the target to reconstruct

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

	Ours			*Transporter* (not comparable)		
# Keypoints	4	8	16	4	8	16
BT-CF	34.40	35.41	34.92	*34.10*	*34.88*	*39.20*
B-CF	37.76	37.06	36.98	*34.75*	*34.78*	*35.13*
C-CF	35.41	34.42	35.98	*32.66*	*33.39*	*34.47*

Table F.4: **Comparison with Transporter** – PSNR (dB) on the task of reconstructing target from the source (both randomly sampled from Filtered-CoPhy), using 5 coefficients per keypoint. We vary the number of keypoints in both our model and the Transporter. Note that Transporter uses target features to reconstruct the image, hence it is not comparable with our model.

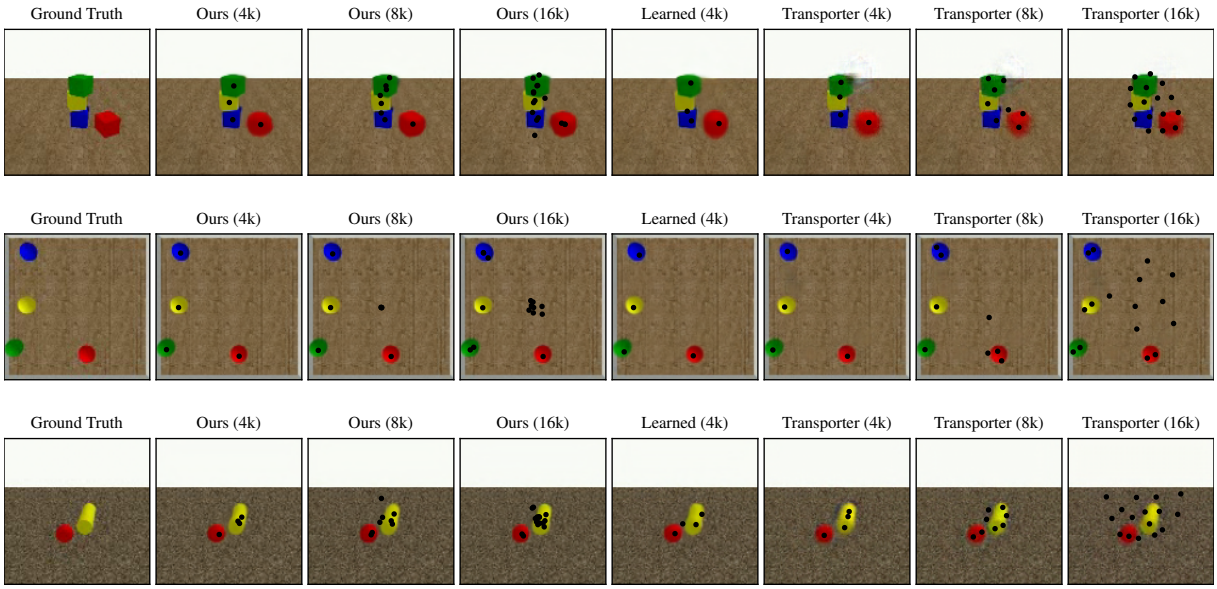


Figure F.6: **Qualitative examples** – Example of the reconstructed image by our-derendering module and the Transporter.

the target. This makes the Transporter incomparable with our de-rendering module. We nevertheless compare the performances of the two models in Table F.6, and provide visual examples in Fig. F.6. Even though the two models are not comparable, as the Transporter uses additional information, our model still outperforms the Transporter for small numbers of keypoints. Interestingly, for higher numbers of keypoints, Transporter tends to discover keypoints far from the object. We investigate this behavior in the following section and show that this is a critical problem for learning causal reasoning on the discovered keypoints.

F.3.2 *Analysis of behavior of Transporter*

The original version of the V-CDN model (Li et al., 2020a) is based on Transporter (Kulkarni et al., 2019). We have already highlighted the fact that this model is not comparable with our task, as it requires not only the target keypoints K_{target} but also a dense feature map f_{target}, whose dynamics can hardly be learned due to its high dimensionality. More precisely, the transport equation (Eq. F-3) allows to pass information from the target by two means: the 2D

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

position of the keypoints (K_{target}) and the dense feature map of the target (f_{target}). The number of keypoints therefore becomes a highly sensible parameter, as the transporter can decide to preferably transfer information through the target features rather than through the keypoint locations. When the number of keypoints is low, they act as a bottleneck, and the model has to carefully discover them to reconstruct the image. On the other hand, when we increase the number of keypoints, the Transporter stops tracking objects in the scene and transfers visual information through the dense feature map, making the predicted keypoints unnecessary for image reconstruction, and therefore not representative of the dynamics.

To illustrate our hypothesis, we set up the following experiment. Starting from a trained Transporter model, we fixed the source image to be X_0 (the first frame from the trajectory) during the evaluation step. Then, we compute features and keypoints on the target frame X_t regularly sampled in time. We reconstruct the target image using the transport equation, *but without updating the target keypoints*. Practically, this consists in computing $\hat{\Psi}_{\text{target}}$ with Eq. (F-3) substituting K_{target} for K_{source}.

Results are shown in Fig. F.7. There is no dynamic forecasting involved in this figure, and the Transporter we used was trained in a regular way, we only changed the transport equation on evaluation time. Even though the keypoint positions have been fixed, the Transporter manages to reconstruct a significant part of the images, which indicates that a part of the dynamics has been encoded in the dense feature map.

In contrast, this issue does not arise from our de-rendering module, since our decoder solely relies on the target keypoints to reconstruct the image. Note that this is not contradictory with the claim in Li et al. (2020a) since they do not evaluate V-CDN in pixel space. A rational choice of the number of keypoints leads to satisfactory performance, allowing V-CDN to accurately forecast the trajectory in keypoints space, and retrieve the hidden confounders on their dataset.

F.3.3 *Temporal inconsistency issues*

Increasing the number of keypoints of the Transporter may lead to temporal inconsistency during the long-range reconstruction. For example, a keypoint that tracks the edge of a cube in the first frame may target a face of this same cube in the future, since dynamics does not intervene in the keypoint discovery process.

Our de-rendering directly addresses this through the usage of additional appearance coefficients, which allows us to limit the number of keypoints to the number of objects in the scene, effectively alleviating the consistency issue. Fig. F.8 illustrates this phenomenon by plotting the discovered keypoint locations forward in time, as well as the 2D location of the center of mass of each object. Note that the Transporter suffers from the temporal inconsistency issue with numbers of keypoints as low as 4 (green cube). In contrast, our model

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

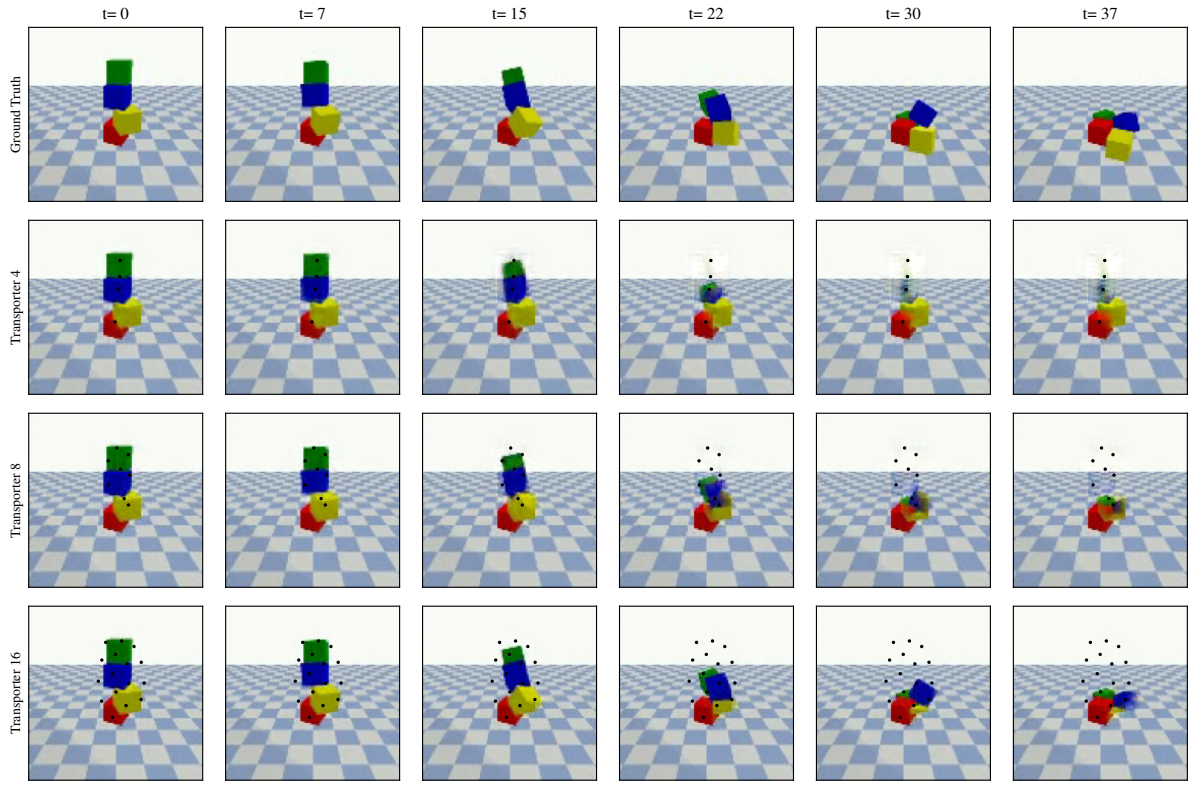


Figure F.7: **Static keypoints** – We evaluate the Transporter with a varying number of keypoints to reconstruct images regularly sampled in a trajectory while having the target keypoints fixed. Even if the keypoints are not moving, the Transporter still manages to reconstruct a significant part of the image, which indicates that the keypoints are not fully responsible for encoding the dynamics of the scene.

manages to solve the problem and accurately tracks the centers of mass, even though they were never supervised.

F.4 Details of model architectures

F.4.1 *De-rendering module*

We call a "block" a 2D convolutional layer followed by a 2D batch norm layer and ReLU activation. The exact architecture of each part of the encoder is described in Table F.5a. The decoder hyper-parameters are described in Table F.5b.

Dense feature map estimator $\mathcal{F}_\theta$ – We compute the feature vector from X_{source} by applying a convolutional network $\mathcal{F}_\theta$ on the output of the common CNN of the encoder. This produces the source feature vector $\boldsymbol{f}_{\text{source}}$ of shape $(\text{batch}, 16, 28, 28)$.

Keypoint detector $\mathcal{K}_\theta$ – is a convolutional network which outputs a set of 2D heatmaps of shape $(\text{batch}, N, 28, 28)$, where N is the desired number of keypoints. We apply a spatial softmax function on the two last dimensions, then we extract a pair of coordinates on

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

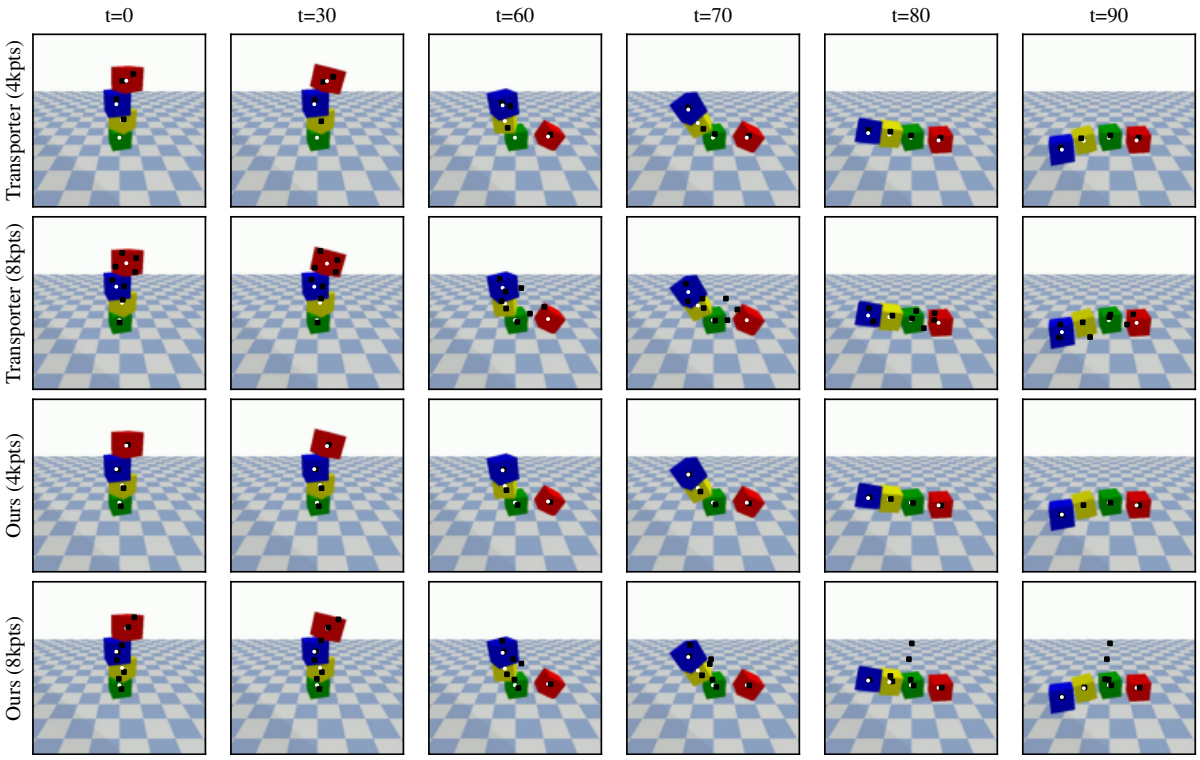


Figure F.8: **Keypoints consistency** – Temporal inconsistency in long-range reconstruction. We show the keypoints discovered on images taken from different time steps (black dots). We also compute the 2D location of the center of mass of each object in the scene (white dots). Our de-rendering module accurately tracks the centers of mass, which have never been supervised.

each heatmap by looking for the location of the maximum, which gives us $\boldsymbol{k}_n^{\text{target}}$ of shape $(\text{batch}, N, 2)$.

Coefficient estimator $\mathcal{C}_\theta$ – We obtain the coefficient by applying a third convolutional network to the output of the common encoder CNN, which again results in a set of 2D vectors of shape $(\text{batch}, N, 28, 28)$. These vectors are flattened channel-wise and provide a tensor of shape $(\text{batch}, N, 28 \times 28))$ fed to an MLP (see Table F.5a for the exact architecture) that estimates the coefficients $\boldsymbol{c}_n^{\text{target}}$ of shape $(\text{batch}, N, C+1)$.

Gaussian mapping $\mathcal{G}$ – The keypoint vector $\boldsymbol{k}_n^{\text{target}}$ is mapped to a 2D vector through a Gaussian mapping process :

$$\mathcal{G}(\boldsymbol{k})(x,y) = \exp\left(-\frac{(x-k_x)^2 + (y-k_y)^2}{\sigma^2}\right), \tag{F-4}$$

where $\mathcal{G}(\boldsymbol{k}) \in \mathbb{R}^{28\times 28}$ is the Gaussian mapping of the keypoint $\boldsymbol{k} = \begin{bmatrix} k_x & k_y \end{bmatrix}$. We deform these Gaussian mappings by applying convolutions with filters H_i controlled by the coefficients $\boldsymbol{c}_n$.

The filters from $\mathcal{H}$ are 5×5 kernels that elongate the Gaussian in a specific direction. Practically, we obtain the filter H_i by drawing a line crossing the center of the kernel and with

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

CNN						
	Module	*in ch.*	*out ch.*	*kernel*	*stride*	*pad.*
1	Block	3	32	7	1	3
2		32	32	3	1	1
3		32	64	3	2	1
4		64	64	3	1	1
5		64	128	3	2	1

$\mathcal{F}_\theta$						
1	Block	128	16	3	1	1

$\mathcal{K}_\theta$						
1	Block	128	128	3	1	1
2	Conv2d	128	N	3	1	1
3	Softplus					

$\mathcal{C}_\theta$						
1	Block	128	N	3	1	1
2	Flatten					
	Module	*in*		*out*		
3	Linear+ReLU	784		2048		
4	Linear+ReLU	2048		1024		
5	Linear+ReLU	1024		512		
6	Linear+ReLU	512		C		
7	Sigmoid					

(a) Encoder architecture

$\mathcal{R}_\theta$						
		in ch.	*out ch.*	*kernel*	*stride*	*pad.*
1	Block	$16+N\times C$	128	3	1	1
2		128	128	3	1	1
3		128	64	3	1	1
4	UpSamplingBilinear2d(2)					
5	Block	64	64	3	1	1
6		64	32	3	1	1
7	UpSamplingBilinear2d(2)					
8	Block	32	32	3	1	1
9		32	32	7	1	1
10	Conv2d	32	3	1	1	1
11	TanH					

(b) Decoder architecture

Table F.5: **Model details** – Architectural details of the de-rendering module.

a slope angle of $i\frac{\pi}{C}$ where C is the number of coefficients. We then apply a 2D convolution :

$$\mathbf{G}_n[i] = \boldsymbol{c}_n^{C+1}\left(\boldsymbol{c}_n^i H_i\right) * \mathcal{G}(\boldsymbol{k}_n). \tag{F-5}$$

Note that we also compute a supplementary coefficient α_n^{C+1} used as a gate on the keypoints. By setting this coefficient to zero, the de-rendering module can deactivate a keypoint (which is redundant with deactivating the full set of coefficients for this keypoint).

Refiner $\mathcal{R}_\theta$ – To reconstruct the target image, we channel-wise stack feature vectors from the source with the constructed filters and feed them to the decoder CNN $\mathcal{R}_\theta$ (Table F.5b).

We trained the de-rendering module on pairs of images $(X_{\text{source}}, X_{\text{target}})$ randomly sampled from sequences D. For a given sequence D, we take $T-25$ first frames of the trajectory as a source (where T is the number of frames in the video). The last 25 frames are used as a target. For evaluation, we take the 25th frame as the source, and the 50th frame as the target. We use Adam optimizer with a learning rate of 10^{-3}, $\gamma_1 = 10^4$ and $\gamma_2 = 10^{-1}$ to minimize (9-4).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

F.4.2 *CoDy*

We describe the architectural choices made in CoDy. Let

$$\boldsymbol{s}[t] = \begin{bmatrix} \boldsymbol{k}_n & \boldsymbol{c}_n & \cdots & \boldsymbol{k}_n & \boldsymbol{c}_n \end{bmatrix} \tag{F-6}$$

be the state representation of an image X_t, composed of the N keypoints 2D coordinates with their $C+1$ coefficients. The time derivative of each component of the state is computed via an implicit Euler derivation scheme $\dot{\boldsymbol{k}}[t] = \boldsymbol{k}[t] - \boldsymbol{k}[t-1]$. We use a subscript notation to distinguish the keypoints from **AB** and CD.

CF estimator – The latent representation of the confounders is discovered from $\boldsymbol{s}^{\mathbf{AB}}$. The graph neural network from this module implements the message passing function g_{edges} and the aggregation function g_{nodes} (see (9-5)) by a MLP with 3 hidden layers of 64 neurons and ReLU activation unit. The resulting nodes embeddings $\boldsymbol{h}^{\mathbf{AB}}[t] = f_{\text{gnn},1}\left(\boldsymbol{s}^{\mathbf{AB}}[t]\right)$ belong to $\mathbb{R}^{128}$. We then apply a GRU with 2 layers and a hidden vector of size 32 to each node in $\boldsymbol{h}^{\mathbf{AB}}[t]$ (sharing parameters between nodes). The last hidden vector is used as the latent representation of the confounders $\tilde{\boldsymbol{z}}_n$.

State encoder-decoder – the state encoder is modeled as a GNN where the message passing function and the aggregation function are MLPs with one hidden layer of 32 units. The encoded state $\sigma^{\text{CD}}[t] = \mathcal{E}\left(s^{\text{CD}}[t]\right)$ lies in $\mathbb{R}^{256}$. We perform dynamical prediction in this σ space and then project back the forecasting in the keypoint space using a decoder. The decoder $\Delta\left(\sigma[t]\right)$ first applies a shared GRU with one layer and a hidden vector size of 256 to each keypoint $\sigma_n[t]$, followed by a GNN with the same structure as the state encoder.

Dynamic system – our dynamic system forecasts the future state $\hat{\sigma}[t+1]$ from the current estimation $\hat{\sigma}[t]$ and the confounders $\tilde{\boldsymbol{z}} = [\tilde{\boldsymbol{z}}_1 \ldots \tilde{\boldsymbol{z}}_N]$. It first applies a GNN to the concatenated vector $[\hat{\sigma}[t]\ \tilde{\boldsymbol{z}}]$. The message passing function and the aggregation function are MLPs with 3 hidden layers of 64 neurons and ReLU activation function. The resulting nodes embeddings $f_{\text{gnn},2}\left(\sigma[t]\right)$ belong to $\mathbb{R}^{64}$ and are fed to a GRU sharing weights among each node with 2 layers and a hidden vector of size 64. This GRU updates the hidden vector $\boldsymbol{v}^{\text{CD}}[t] = [\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_N]$, that is then used to compute a displacement vector with a linear transformation:

$$\hat{\sigma}_n[t+1] = \sigma_n[t] + W\boldsymbol{v}_n[t] + \boldsymbol{b}. \tag{F-7}$$

CoDy is trained using Adam to minimize (9-8) (learning rate 10^{-4}, $\gamma_3 = 1$). We train each CoDy instance by providing it with fixed keypoints states $\boldsymbol{s}^{\mathbf{AB}}[t]$ and the initial condition $\boldsymbol{s}^{\text{CD}}[0]$ computed by our trained de-rendering module. CoDy first computes the latent confounder representation $\tilde{\boldsymbol{z}}$, and then projects the initial condition into the latent dynamic space $\sigma[0] = \mathcal{E}\left(\boldsymbol{s}^{\text{CD}}[0]\right)$. We apply the dynamic model multiple times to recursively forecast T time steps from CD. We then apply the decoder $\Delta\left(\hat{\sigma}[t]\right)$ to compute the trajectory in the keypoint space.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

F.5 Additional quantitative evaluation

F.5.1 *Multi-object tracking metrics*

When the number of keypoints matches the number of objects in the scene, the keypoint detector naturally and in an unsupervised manner places keypoints near the center of mass of each object (see Figure F.8). Leveraging this emerged property, we provide additional empirical demonstration of the accuracy of our model by computing classical Multi-Object Tracking (MOT) metrics. In particular, we computed the Multi-Object Tracking Precision (MOTP) and the Multi-Object Tracking (MOTA) as described in Bernardin and Stiefelhagen (2008).

- **MOTA** requires computing the number of missed objects (i.e. not tracked by a keypoint) and the number of false positives (i.e. keypoints that do not represent an actual object). MOTA takes values in $[-1, 1]$, where 1 represents perfect tracking:

$$\text{MOTA} = 1 - \frac{\sum_t m_t + f_t + s_t}{\sum_t g_t}, \tag{F-8}$$

  where m_t is the number of missed objects at time t, f_t is the number of false positives at time t, s_t is the number of swaps at time t, and g_t is the number of objects at time t.
- **MOTP** is a measurement of the distance between the keypoints and the ground-truth centers of mass conditioned on the pairing process:

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}, \tag{F-9}$$

  where c_t is the number of accurately tracked objects at time t and d_t^i is the distance between the keypoint and the center of mass of the ith association {keypoints+center of mass}.

Note that these metrics are related: low MOTP indicates that the tracked objects are tracked precisely, and low MOTA indicates that many objects are missed. Thus, to be efficient, a model needs to achieve both low MOTP and high MOTA.

We also reported the performances of CoPhyNet (Baradel et al., 2020) that predicts counterfactual outcomes in Euclidian space using the ground-truth 3D space. As it uses ground truth object positions during training, it is not comparable and should be considered as a soft upper bound of our method. We present our results in Table F.6. This confirms the superiority of our method over UV-CDN in keypoint space. The upper bound CoPhyNet takes advantage of the non-ambiguous 3D representation modeled by the ground-truth state of the object of the scene.

Our method also outperforms CoPhyNet on the `ballsCF` task, probably due to two phenomena. First, `ballsCF` is the only 2D task of FilteredCoPhy. Thus, CoPhyNet does not have the advantage of using ground-truth 3D positions. Second, the state-encoder in CoDy projects

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

		Ours	UV-CDN	*CoPhyNet (not comparable)*
BT-CF	MOTA ↑	0.46	0.16	*0.44*
	MOTP ↓	3.34	4.51	*0.72*
B-CF	MOTA ↑	-0.07	-0.73	*-0.16*
	MOTP ↓	4.64	5.83	*5.10*
C-CF	MOTA ↑	-0.14	-0.19	*0.21*
	MOTP ↓	6.35	6.35	*4.37*

Table F.6: **MOT metrics** – for different methods. While not comparable, we report the CoPhyNet performance as a soft upper bound. Our method and UV-CDN use one keypoint per object. MOTA ↑: higher is better; MOTP ↓: lower is better;

the 2D position of each sphere in a space where the dynamics is easier to learn, probably by breaking the non-linearity of collisions.

F.5.2 *Impact of the do-operations*

We also measure the impact of the do-operation types on video forecasting. Fig. F.9 (left) is obtained by computing PSNR for each example of the training set and reporting the result on a 2D graph, depending on the amplitude of the displacement that characterizes the do-operation. We applied the same method to obtain Fig.F.9 (right) that focuses on the type of do-operation, that is moving, removing, or rotating an object. These figures are computed using the 2K keypoints models.

Our method generalizes well across different do-operations, including both the type of the operation and the amplitude. A key to this success is the careful design of the dataset (balanced with respect to the types of do-operations), and a reasonable representation (our set of keypoints and coefficients) able to detect and model each do-operation from images.

F.6 Experiments on real-world data

Our contributions are focused on the discovery of causality in physics through counterfactual reasoning. We designed our model to solve the new benchmark and provided empirical evidence that our method is well suited for modeling rigid-body physics and counterfactual reasoning. The following section aims to demonstrate that our approach can also be extended to a real-world dataset. We provide qualitative results obtained on a derivative of `BlocktowerCF` using real cubes tower (Lerer et al., 2016).

We refer to this dataset as *Blocktower IRL*. It is composed of 516 videos of wooden blocks stacked in a stable or unstable manner. The amount of cubes in a tower varies from 2 to 4. We aim to predict the dynamics of the tower in pixel space. This is highly related to our task `BlocktowerCF` (which was inspired by the seminal work from Lerer et al. (2016)) with three main differences: (1) the dataset shows real cube towers, (2) the problem is not counterfactual,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

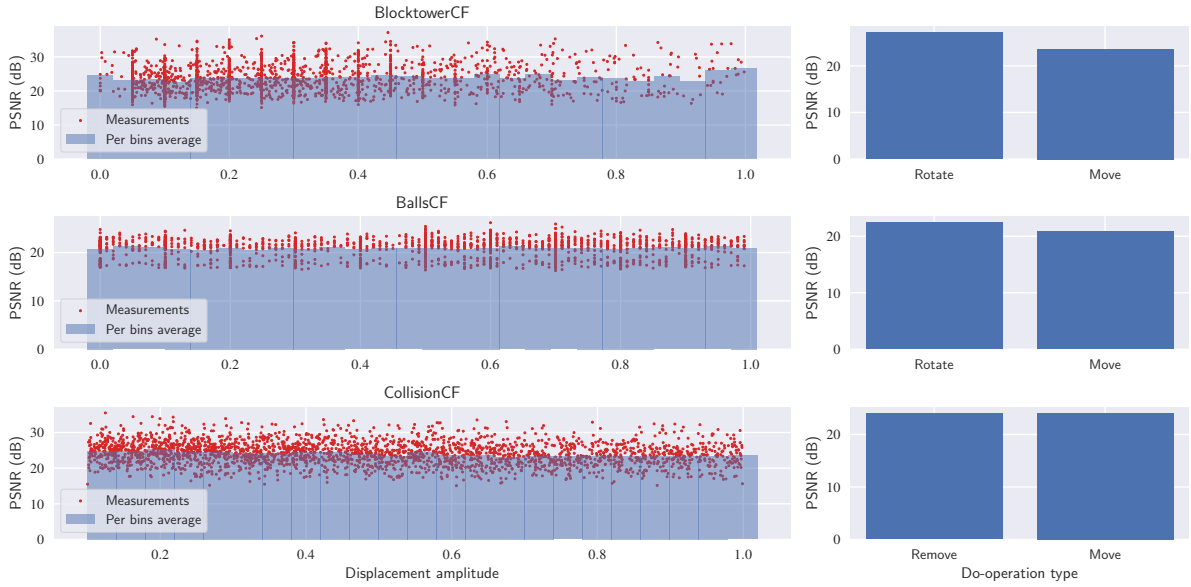


Figure F.9: **Effect of the do-operation** – on the quality of the forecasted video. (left) our method generalizes well to a wide range of "Move" operation amplitudes. (right) We observe a difference of 3dB in favor of the *Move* do-operation, which is unsurprising, as it is the least disturbing intervention

i.e. every cube has the same mass and (3) the dataset contains only a few videos.

To cope with the lack of data, we exploit our pre-trained models on `BlocktowerCF` and fine-tune them on Blocktower IRL. The adaptation of the de-rendering module is straightforward: we choose the 4 keypoints-5 coefficients configuration and train the module for image reconstruction after loading the weights from previous training on our simulated task. CoDy, on the other hand, requires careful tuning to preserve the learned regularities from `BlocktowerCF` and prevent over-fitting. Since Blocktower IRL is not counterfactual, we deactivate the confounder estimator and set $\tilde{z}_n$ to vectors of ones. We also freeze the weights of the last layers of the MLPs in the dynamic model.

To the best of our knowledge, we are the first to use this dataset for video prediction. Lerer et al. (2016) and Wu et al. (2017a) leverage the video for stability prediction but actual trajectory forecasting was not the main objective. To quantitatively evaluate our method, we predict 20 frames in the future from a single image sampled in the trajectory. We measured an average PSNR of 26.27 dB, which is of the same order of magnitude compared to the results obtained in simulation. Figure F.10 provides a visual example of the output.

F.7 Qualitative evaluation: more visual examples

More qualitative results produced by our model on different tasks from our datasets are given below.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

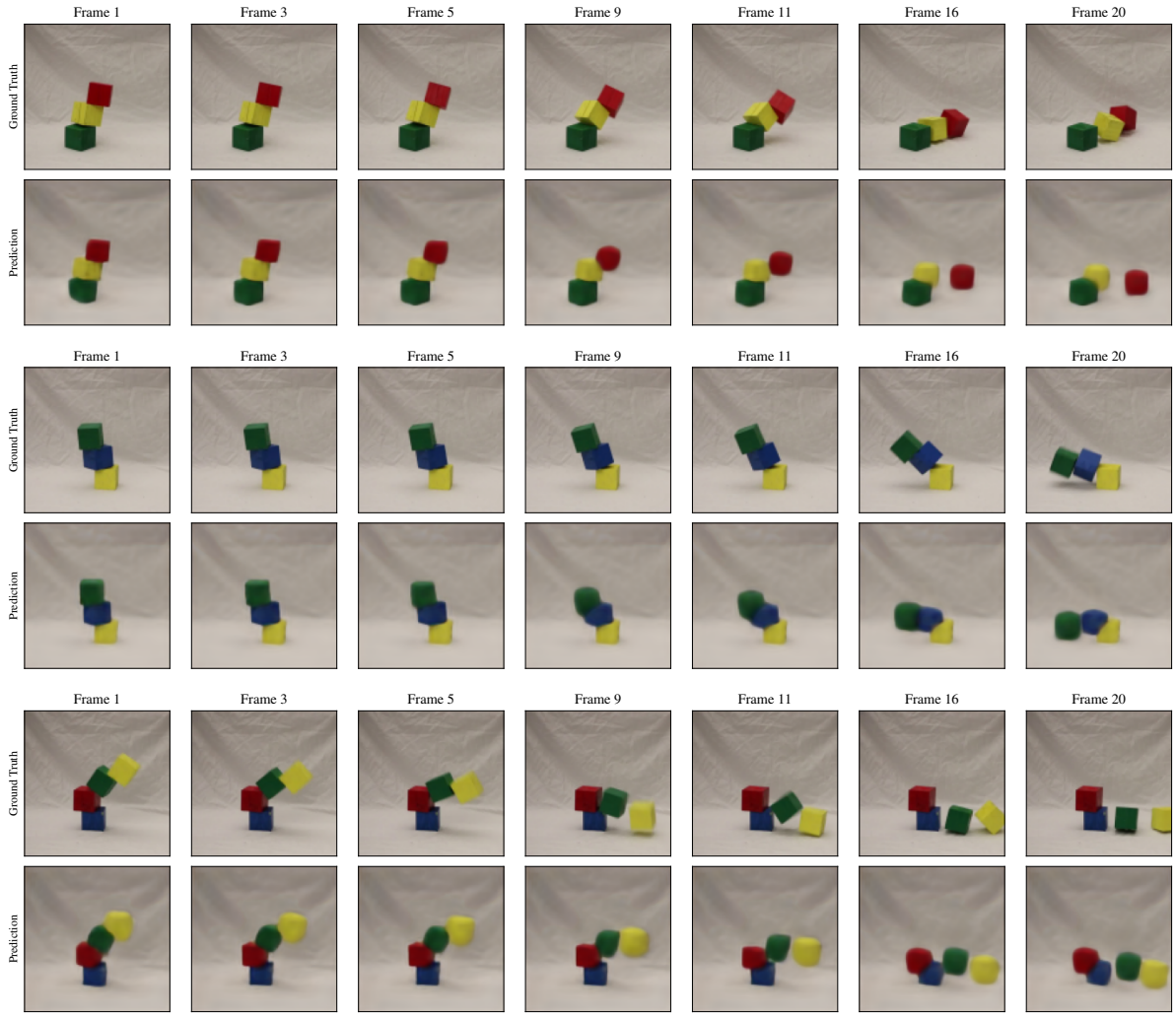


Figure F.10: **Real-world dataset** – We evaluate our method on a real-world dataset Blocktower IRL. After fine-tuning, CoDy manages to accurately forecast future frames from real videos.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

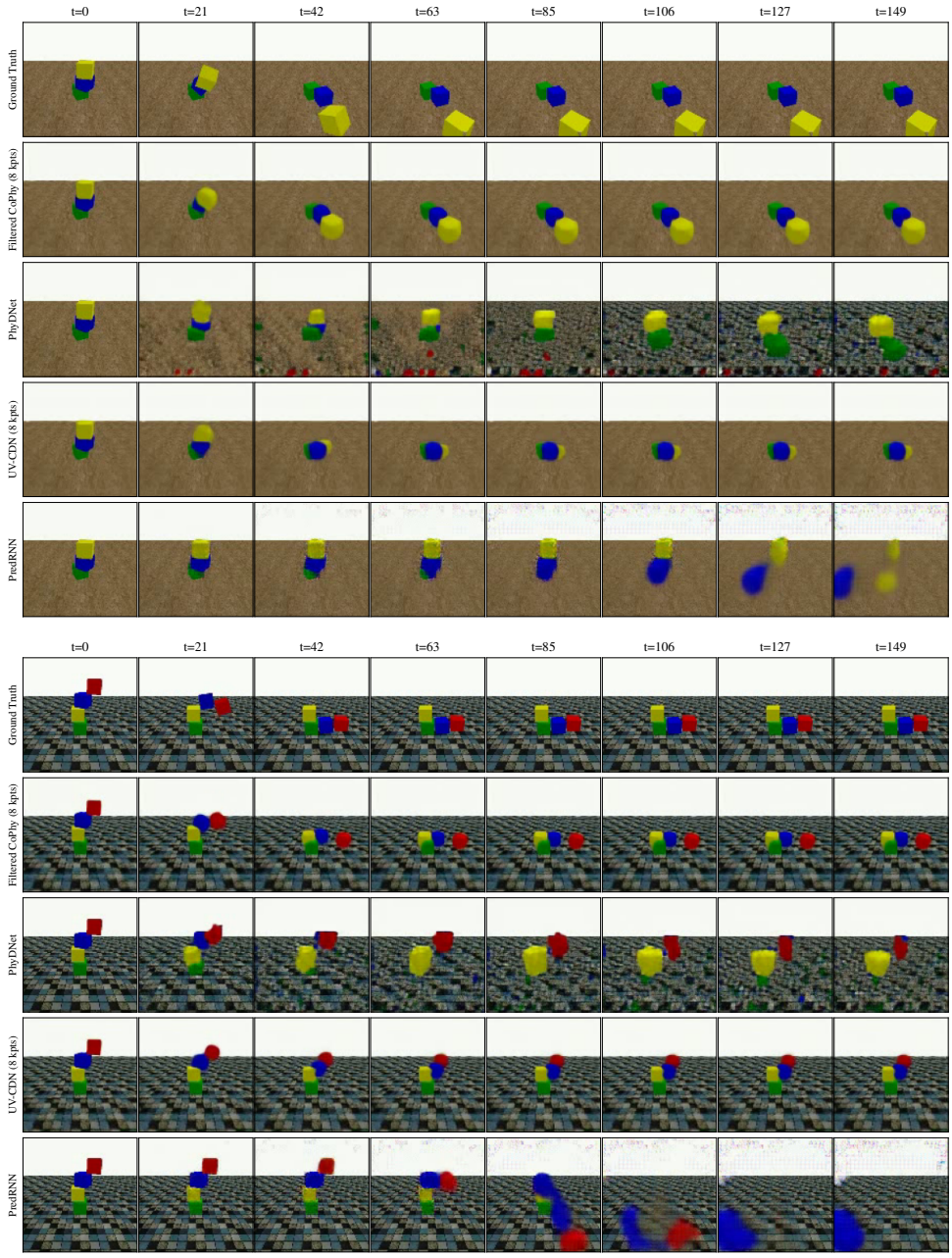


Figure F.11: Qualitative performance on the `BlocktowerCF` (BT-CF) benchmark.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

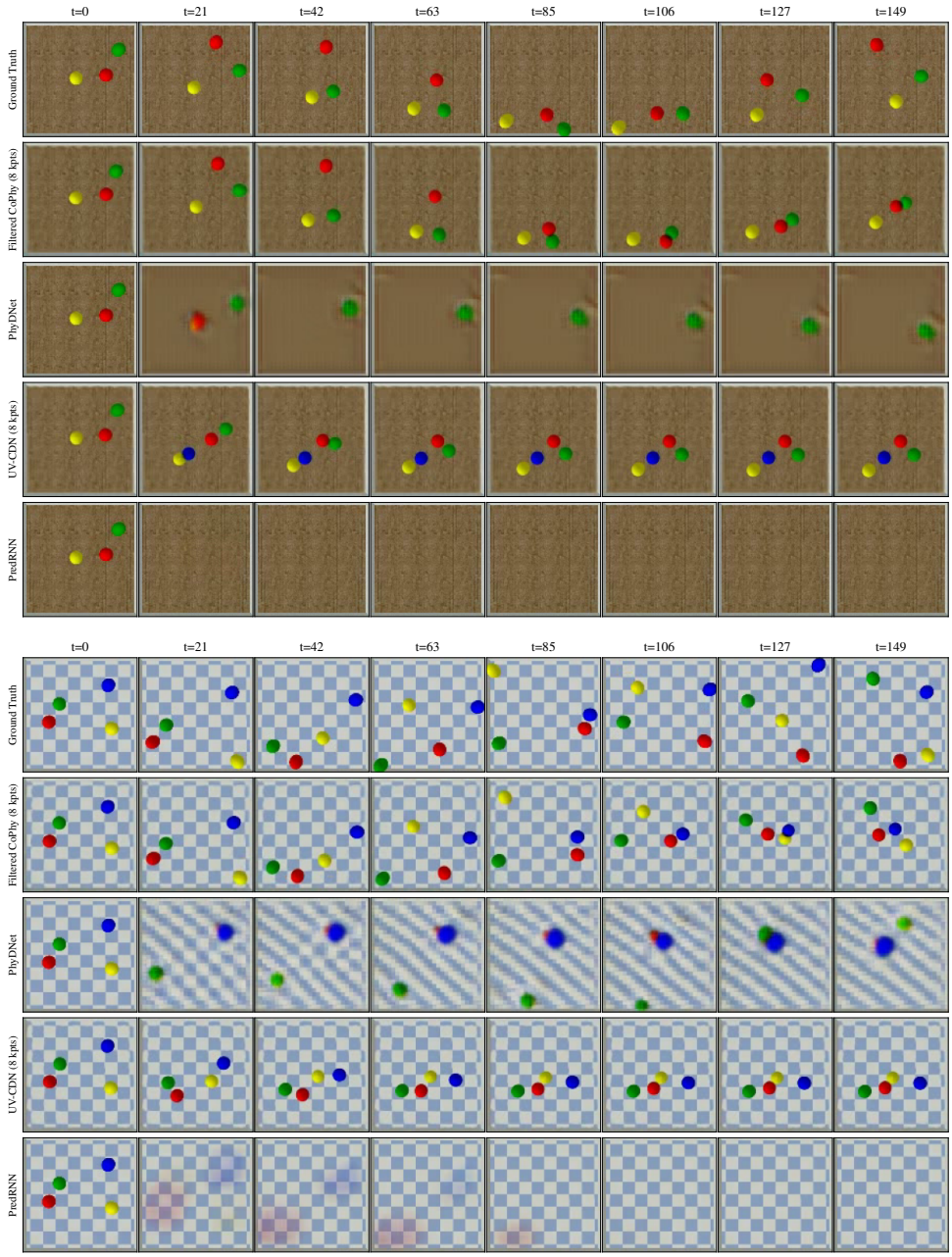


Figure F.12: Qualitative performance on the `BallsCF` (B-CF) benchmark.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

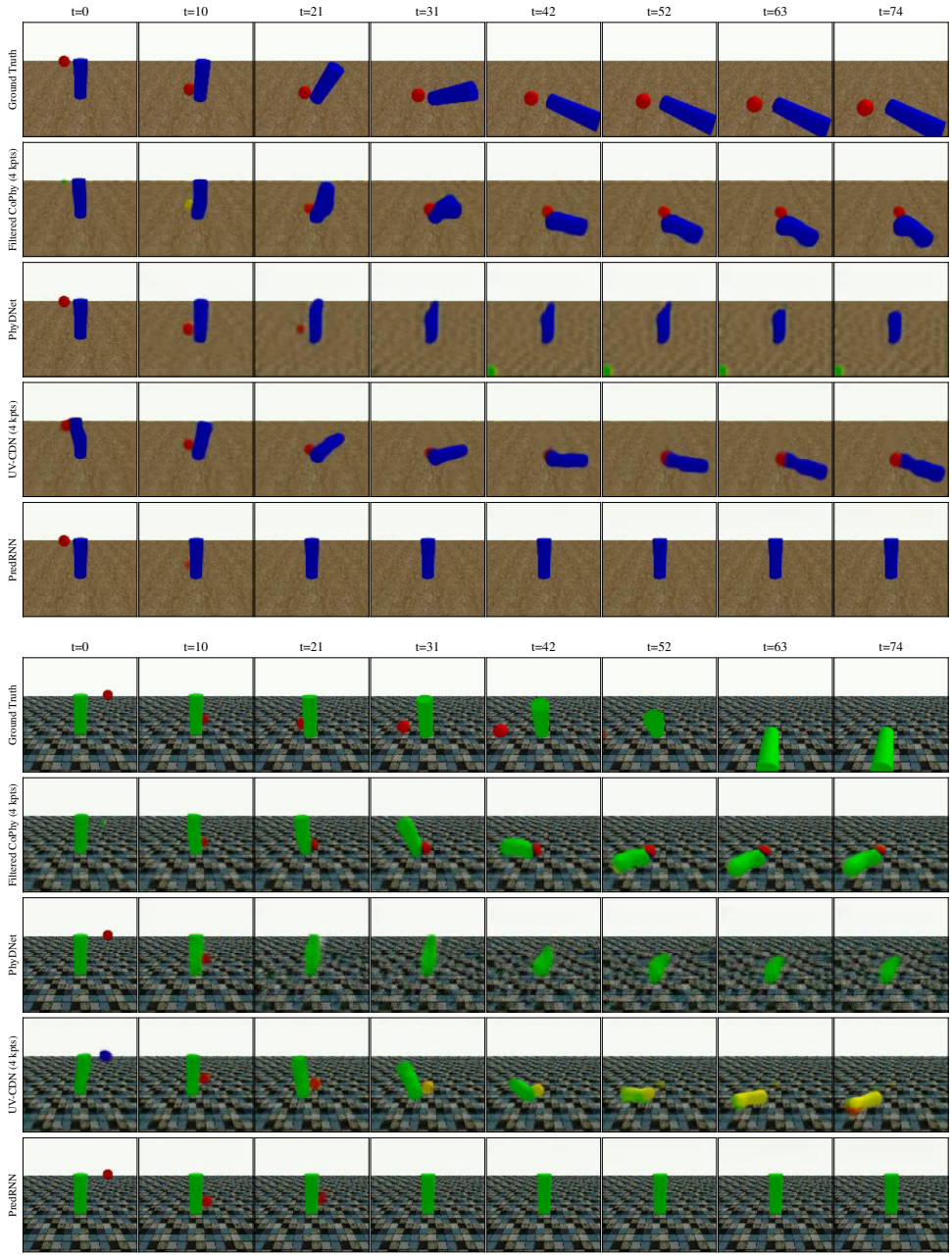


Figure F.13: Qualitative performance on the `CollisionCF` (C-CF) benchmark.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés

FOLIO ADMINISTRATIF

THESE DE L'INSA LYON, MEMBRE DE L'UNIVERISTE DE LYON

NOM : Janny DATE de SOUTENANCE: 16 Janvier 2023

Prénoms : Steeven

TITRE:

Identification and Simulation of Physical Systems with Structured Deep Learning and Inductive Knowledge

NATURE: Doctorat Numéro d'ordre : 2024ISAL0001

Ecole doctoral: infomath

Spécialité : informatique

Resumé : Les progrès technologiques de notre époque sont soutenus par la disponibilité croissante d'outils numériques pour simuler, contrôler et observer les systèmes physiques. En se concentrant sur des phénomènes de plus en plus complexes, nos outils conventionnels ne parviennent pas à répondre aux attentes croissantes des ingénieurs, que ce soit en termes de précision ou de temps de calcul. Les approches data-driven, en particulier les réseaux de neurones, offrent des alternatives prometteuses pour résoudre ces problèmes. Ces types de modèles capturent des relations complexes et non linéaires dans les systèmes physiques et déplacent la charge de modélisation vers celle de la collecte de données. Cependant, ces nouvelles méthodes sacrifient souvent les critères de stabilité, de robustesse et de précision et plus généralement les garanties offertes par les approches traditionnelles. Nous proposons de combiner les domaines de la physique, de l'apprentissage profond et de la théorie du contrôle pour proposer de nouvelles méthodes hybrides, tirant parti de la puissance des réseaux de neurones, tout en s'appuyant sur des biais inductifs issus de la physique. Ce manuscrit présente nos travaux dans ce domaine. En particulier, il décrit des outils théoriques (abordés dans la partie 1) liés à la simulation de systèmes dynamiques et les connecte à la conception de réseaux neuronaux. Dans un deuxième temps (Partie 2), nous exploitons ces connaissances pour concevoir des algorithmes de contrôle et des techniques de simulation impliquant la résolution de problèmes complexes liés aux équations aux dérivées partielles. Enfin, dans la troisième partie, nous abordons des problèmes de simulation à plus grande échelle tels que la dynamique des fluides et le raisonnement contrefactuel. Nos travaux ont été présentés lors de conférences scientifiques dans le domaine de l'intelligence artificielle et de la théorie du contrôle. En construisant un pont entre la physique et l'apprentissage automatique, nous croyons fermement que cette direction de recherche peut contribuer à une nouvelle génération de méthodologies pour la simulation et le contrôle des systèmes physiques.

MOTS-CLES: Deep learning, réseaux de neurones, controle, physique

Laboratoire(s) de rechercher : LIRIS, LAGEPP

Directeur de thèse :DIGNE Julie

Président du jury :

Composition du jury :

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0001/these.pdf
© [S. Janny], [2024], INSA Lyon, tous droits réservés