Experimental data assimilation : learning-based estimation for state-space models

Mona Buisson-Fenet

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Mines Paris – PSL

Experimental data assimilation: learning-based estimation for state-space models

Assimilation de données expérimentales : estimation à base d'apprentissage pour modèles sous forme d'état

Soutenue par

Mona Buisson-Fenet

Le 27 avril 2023

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité

Mathématique et Automatique

Composition du jury :

David RYCKELYNCK Mines Paris	*Président du jury*
Ioannis STEFANOU École Centrale de Nantes	*Rapporteur*
Bahman GHARESIFARD University of California, Los Angeles	*Rapporteur*
Madiha NADRI Université Claude Bernard Lyon 1	*Examinateur*
Valéry MORGENTHALER Ansys Inc	*Examinateur*
Sebastian TRIMPE RWTH Aachen University	*Directeur de thèse*
Florent DI MEGLIO Mines Paris	*Directeur de thèse*

MINES PARIS | PSL

MINES PARIS

DOCTORAL THESIS

Experimental data assimilation: learning-based estimation for state-space models

Author:
Mona BUISSON-FENET

Supervisors:
Florent DI MEGLIO
Sebastian TRIMPE
Valéry MORGENTHALER

Centre Automatique et Systèmes - Mines Paris - PSL
Institute for Data Science in Mechanical Engineering - RWTH Aachen University
Ansys France Research Team

Résumé

La modélisation et la simulation numérique de processus complexes sont aujourd'hui des éléments essentiels du développement industriel. De par la récente augmentation des capacités de génération, de collecte et de traitement des données, les méthodes basées sur l'apprentissage apparaissent aujourd'hui comme un complément prometteur à la modélisation physique. Unir ces deux points de vue permettrait notamment de créer des jumeaux digitaux : des reproductions numériques exactes d'objets physiques combinant un modèle de simulation haute-fidélité et des données expérimentales recueillies sur le système réel. Cependant, les données disponibles sur les plateformes physiques sont généralement bruitées et tous les états ne peuvent pas être mesurés. Notre objectif est *d'extraire des informations de ces données expérimentales sur le modèle sous forme d'état sous-jacent*. Pour ce faire, nous combinons des concepts issus du design d'observateurs non linéaires et du machine learning.

Dans une première partie, nous supposons que le système doit être identifié à partir des données : cette information prend la forme d'un modèle dynamique. En raison de la nature partielle des observations et d'un modèle imparfait, il est nécessaire d'estimer conjointement l'état sous-jacent et sa dynamique. Nous commençons par supposer que le système est dans la forme canonique observable, ce qui permet de construire un observateur grand gain, et de l'interconnecter avec un processus gaussien approximant la dynamique. La convergence conjointe de l'état estimé et de la dynamique peut alors être prouvée pour cette combinaison. Nous étendons ensuite ces idées à des systèmes plus généraux en combinant les équations différentielles ordinaires neuronales avec un modèle basé sur un observateur, qui fait correspondre les observations à l'état latent initial.

Dans une deuxième partie, nous nous éloignons de la motivation initiale des jumeaux numériques et supposons que cette information porte sur l'estimation d'état elle-même. Nous commençons par définir la distinguabilité pour les systèmes stochastiques non linéaires, puis nous la quantifions à partir de données de sortie à l'aide de méthodes statistiques. Cela permet d'évaluer dans quelle mesure les états initiaux et les configurations des capteurs peuvent être distingués à partir de ces données. Nous utilisons ensuite des réseaux de neurones pour approximer une transformation permettant le design d'observateurs de Kazantis-Kravaris / Luenberger numériques, et nous proposons un critère empirique qui guide le réglage de l'observateur.

Abstract

Numerical simulation and modeling of complex processes is a critical part of industrial development. With the recent increase in data generation, collection, and processing capabilities, learning-based methods appear as a promising addition to physics-based modeling. Uniting both views is an appealing prospect, e.g., to create digital twins: exact numerical replicas of physical objects combining a high-fidelity simulation model with experimental data gathered on the real system. However, the data available from physical platforms is typically imperfect; for example, it may be noisy and not cover all state variables. Our aim is to *extract information from this experimental data about the underlying state-space model that explains it*. For this, we combine concepts from nonlinear observer design and modern machine learning.

In a first part, we assume the system needs to be identified from data: this information takes the form of a dynamics model. Due to the partial nature of the observations and an imperfect model, it is necessary to jointly estimate the underlying latent state and its dynamics. We start by assuming the system is in the observable canonical form, allowing to build a high-gain observer, and interconnect it with a Gaussian process dynamics model. We prove joint convergence of the state and dynamics estimates for this combination. We then extend these ideas to more general systems by coupling neural ordinary differential equations with an observer-based recognition model that maps the observations to the initial latent state.

In a second part, we shift the focus away from the initial motivation of digital twins and assume this information relates to state estimation itself. We start by defining distinguishability for nonlinear stochastic systems, then quantify it given a set of output samples using statistical methods. This allows for assessing how distinguishable initial states and sensor configurations are given this data. We then make use of neural networks to approximate a transformation enabling general nonlinear observer design with Kazantis-Kravaris / Luenberger observers, and derive an empirical criterion that guides the tuning of the observer.

Contents

List of Figures

Abbreviations

ROM	Reduced Order Model
GP	Gaussian Process
GPSSM	Gaussian Process State-Space Model
NN	Neural Network
PINN	Physics-Informed Neural Network
AE	AutoEncoder
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gatred Recurrent Unit
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
SDE	Stochastic Differential Equation
NODE	Neural Ordinary Differential Equation
RKHS	Reproducing Kernel Hilbert Space
KME	Kernel Mean Embedding
MMD	Maximum Mean Discrepancy
KRR	Kernel Ridge Regression
RV	Random Variable
KF	Kalman Filter
EKF	Extended Kalman Filter
KKL	Kazantis-Kravaris / Luenberger
HGO	High-Gain Observer
RMSE	Root Mean Squared Error
RK4	Explicit Runge-Kutta method of order 4
RK4/5	Explicit Runge-Kutta method of order 5(4) [1]
SVD	Singular Value Decomposition
LTI	Linear Time-Invariant
PBN	Probabilistic Boolean Network
LHS	Latin Hypercube Sampling
resp.	respectively
i.i.d.	independent and identically distributed
i.e.,	id est
e.g.,	exempli gratia
w.r.t.	with respect to
iff.	if and only if

Chapter 1
Introduction - version française

La simulation numérique est un élément essentiel du développement industriel. Elle permet de tester différents choix de conception et d'évaluer des stratégies plus rapidement et à moindre coût que le prototypage standard. Certaines entreprises, dont Ansys, ont contribué à ce développement en produisant des simulateurs haute fidélité qui prédisent avec précision le comportement des systèmes physiques. De nombreux domaines ou industries s'appuient sur les simulateurs d'Ansys pour examiner des projets futurs, qu'il s'agisse de simulations de champs d'écoulement pour l'industrie aéronautique ou de modèles de moteurs magnétiques pour les voitures électriques. Malgré cela, la simulation numérique est encore confrontée à de nombreux défis, tels que l'augmentation de la vitesse de calcul ou l'amélioration des modèles, rendus imprécis par des phénomènes non modélisés ou des paramètres dont la valeur réelle ne peut qu'être approximée.

La multiplication des dispositifs de collecte de données et des capacités de calcul sur les plateformes physiques ouvre des possibilités supplémentaires : comment l'industrie peut-elle tirer parti des capacités de la simulation numérique, tout en profitant des données expérimentales qu'elle génère ? De telles interrogations ont conduit au concept de *jumeaux digitaux* [2]–[5]. Les jumeaux digitaux visent à créer une réplique numérique exacte d'un objet physique, en combinant un modèle de simulation haute fidélité avec des données expérimentales enregistrées sur le système. La première étape consiste à construire des modèles de simulation fonctionnant en temps quasi-réel. En effet, les simulateurs haute fidélité décrivant un processus physique complexe peuvent nécessiter plusieurs jours ou semaines de calcul, de sorte que des mises à jour incrémentelles du modèle en fonction des observations peuvent s'avérer irréalisables. La question de la fusion des données expérimentales dans le modèle de simulation peut ensuite être abordée.

Ansys travaille à l'extension de sa suite logicielle pour inclure de tels projets. Pour relever le défi de la vitesse de calcul, Ansys développe un logiciel interne de modélisation d'ordre réduit et d'identification de systèmes non linéaires appelé DynaROM, qui apprend un modèle d'ordre réduit ("reduced order model", ROM) à partir de scénarios d'entraînement générés par des simulations haute fidélité. En particulier, la dimension de l'état latent du ROM est déduite en le construisant itérativement jusqu'à ce qu'une précision souhaitée soit atteinte, et de nombreux autres hyperparamètres peuvent être choisis automatiquement. Ce ROM peut alors prédire des comportements complexes tels que la fatigue des grandes turbines à gaz [6] ou l'évolution spécifique au patient des dissections aortiques [7] en temps proche du temps réel. D'une manière générale, la construction de modèles à la fois précis et légers est un défi majeur pour la création de jumeaux numériques. De nombreuses idées de recherche émergent pour y parvenir, comme l'utilisation de classificateurs pour choisir entre des ROM locaux [5], [8], [9] ou le remplacement de parties de simulateurs numériques par des modèles d'apprentissage profond formés à partir de données de simulation comme dans DynaROM [10]. Ces nouvelles directions de recherche motivent cette thèse.

La question de l'analyse des données expérimentales des systèmes dynamiques intéresse différentes communautés. Dans ce travail, nous proposons de combiner des résultats issus de la théorie du contrôle d'une part, et la puissance de régression de l'apprentissage automatique moderne d'autre part. Nous proposons de nouvelles méthodes à cette intersection. La collecte de données sur les systèmes complexes est coûteuse et toutes les variables du système ne peuvent pas être mesurées. Par conséquent, les principaux défis techniques résident dans la nature partielle et bruitée des observations, associée à un modèle imparfait. L'analyse doit donc permettre d'estimer à la fois la dynamique et l'état du système. Nous supposons que les systèmes physiques en question peuvent être représentés par des modèles sous forme d'état ("state-space model", SSM), car il s'agit d'un point de vue bien établi pour lequel il existe de nombreux résultats à la fois en théorie du contrôle et en apprentissage automatique, étant donné qu'il rend l'état sous-jacent explicite. Notre objectif est alors de *développer des méthodes génériques pour extraire des informations de données expérimentales provenant d'un modèle d'espace d'état sous-jacent*.

1.1 Travaux connexes

À cette fin, nous intégrons des concepts issus de l'apprentissage automatique et de la théorie du contrôle. Dans cette section, nous donnons un bref aperçu de l'état de l'art sur ce sujet, dans les deux communautés et à leur interface. Nous faisons la distinction entre l'identification du système d'une part, c'est-à-dire la détermination de la dynamique à partir d'observations, et l'estimation de l'état d'autre part, c'est-à-dire l'inférence de l'état latent. Nous soulignons les lacunes présentes dans chaque direction de recherche et insérons nos contributions dans cette vue d'ensemble. Nous nous concentrons principalement sur les travaux qui sont pertinents pour cette thèse ; une introduction plus technique aux travaux les plus appropriés est donnée au Ch. 3. Les travaux directement reliés sont discutés plus en détail dans chaque chapitre correspondant.

L'apprentissage automatique a récemment eu beaucoup de succès dans de nombreuses tâches statiques telles que le traitement d'images ou la génération de discours. Toutefois, des méthodes capables de s'attaquer à la complexité, aux autocorrélations et au manque de données du monde physique restent à découvrir. C'est pourquoi il existe une communauté croissante à l'interface entre apprentissage automatique et théorie des systèmes. Cette communauté mène des recherches sur de nombreux sujets, comme l'apprentissage par renforcement [11], [12] ou le contrôle basé sur l'apprentissage [13], [14]. Le sujet de l'identification des systèmes se situe naturellement à cette interface, puisqu'il vise à déduire les inconnues d'un SSM sous-jacent à partir de mesures. Ces inconnues peuvent par exemple être le nombre d'états, les paramètres d'un modèle analytique ou des parties entières du modèle. De nombreuses méthodes ont été développées dans la littérature classique de la théorie du contrôle pour traiter l'incertitude portant sur ces paramètres. Pour le cas linéaire, les "subspace methods" visant à estimer les matrices du système sont bien établies [15]. Il n'existe pas de vision unifiée pour les systèmes non linéaires [16], [17], mais de nombreuses approches parmi lesquelles choisir en fonction du cas d'usage en question. Celles-ci vont des méthodes basées sur l'optimisation qui estiment les paramètres inconnus d'un modèle donné en utilisant les moindres carrés ou des fonctions de coût apparentées [18]–[20], aux méthodes bayésiennes qui se concentrent sur les estimations probabilistes ou a posteriori [21], [22], ou aux modèles boîte noire qui approximent les parties inconnues du modèle sur une base de fonctions appropriée [23].

Cependant, la plupart de ces méthodes classiques sont spécifiques à un système et nécessitent des connaissances spécifiques. C'est pourquoi les travaux récents s'appuient sur l'apprentissage automatique moderne pour élaborer des algorithmes d'identification de systèmes plus généraux. Par exemple, les modèles de processus gaussien sous forme d'état ("Gaussian process state-space models", GPSSM) [24], [25] ont obtenu des résultats prometteurs grâce à leur efficacité d'échantillonnage et à leur formulation analytique, qui permet notamment des garanties [26], [27]. Cependant, l'apprentissage des GPSSM à partir d'observations partielles reste un problème ouvert, et la plupart des approches existantes reposent sur une optimisation non convexe lourde et n'offrent pas de garanties d'approximation. Ce point est abordé dans la Sec. 3.1 après une introduction plus technique. Nous étudions l'apprentissage de tels modèles à partir de mesures partielles dans le Ch. 4.

Une autre ligne de travail se concentre sur l'utilisation de réseaux de neurones profonds pour apprendre des modèles de dynamique. Par exemple, l'identification sparse dans une bibliothèque de lois physiques [28], [29], les réseaux de neurones récurrents ("recurrent neural networks", RNN) [30], [31], ou les équations différentielles ordinaires neuronales ("neural ordinary differential equations", NODE) [32], [33] ont été utilisés pour identifier des systèmes dynamiques. Ces techniques reposent souvent sur une transformation de la sortie vers les coordonnées latentes dans lesquelles la dynamique est apprise, transformation qui peut prendre la forme d'un autoencodeur [28], [34]. La question générale de l'application des connaissances physiques dans les modèles d'apprentissage profond a également suscité un intérêt significatif [4], [35]–[37]. Néanmoins, des approches générales mais qui peuvent être adaptées aux connaissances physiques disponibles et qui peuvent apprendre à partir d'observations partielles restent à développer. Nous construisons une telle approche basée sur les NODE dans le chapitre 5. Un résumé de cette littérature est fourni dans la Sec. 3.2. Inversement, les méthodes d'apprentissage automatique peuvent également bénéficier de l'éclairage de la théorie du contrôle. Par exemple, les capacités d'approximation des RNN [38] ou des NODE [39] ont été étudiées par le biais du contrôle optimal.

L'estimation d'état se situe également à l'interface entre la modélisation sous forme d'état et l'analyse des données, car elle vise à déduire l'état latent complet d'un SSM à partir d'observations partielles. C'est pourquoi des tentatives ont été faites pour exploiter les outils d'apprentissage automatique afin de résoudre des tâches d'estimation d'état [40], [41]. Dans la littérature classique de théorie du contrôle, on parle également de design d'bservateurs. Bien qu'il n'existe pas d'observateur universel établi pour les systèmes non linéaires [42], [43], certains modèles généraux existent, tels que les observateurs grand gain [44], les filtres de Kalman étendus ("extended Kalman filter", EKF) [45], ou les observateurs de Kazantis-Kravaris / Luenberger (KKL) [46], [47]. Plus de détails sont donnés dans la Sec. 3.4. Ces approches peuvent aider à extraire des informations d'observations partielles et à préconditionner les problèmes d'apprentissage correspondants. Par ailleurs, les outils modernes de régression non linéaire peuvent fournir une approximation de certaines fonctions difficiles à déterminer analytiquement dans les résultats classiques, même lorsqu'une transformation en coordonnées canoniques est nécessaire, comme pour les observateurs grand gain ou les observateurs KKL. Exploiter l'apprentissage automatique a donc le potentiel de rendre ces observateurs universels plus facilement applicables en pratique. Nous explorons cette direction dans le Ch. 7.

Enfin, des outils statistiques tels que les méthodes à noyaux [48] peuvent également servir à analyser les données expérimentales. Ils ont plutôt été utilisés comme

outils de régression pour identifier des systèmes dynamiques, mais peuvent également bénéficier d'applications plus larges en examinant des objets plus complexes tels que les distributions de probabilité. Ceci permet de comparer les mesures des systèmes stochastiques et ce qu'elles révèlent sur le SSM sous-jacent. Par exemple, la propriété d'observabilité est difficile à définir et encore plus à vérifier pour les systèmes stochastiques non linéaires ; l'évaluation statistique des données de sortie permet de faire un pas dans cette direction, comme cela est étudié dans le Ch. 6. Un bref aperçu de ces méthodes est donné dans la Sec. 3.3.

1.2 Structure de la thèse

Dans cette thèse, nous développons des méthodes génériques pour extraire des informations des données expérimentales générées par un système dynamique. Chaque méthode est développée dans un contexte spécifique, pour un sous-problème spécifique, et correspond à l'un des articles listés ci-dessous. Une introduction technique est fournie dans Ch. 3, puis les principales contributions sont détaillées dans Ch. 4–7.

Nous examinons deux axes de travail. Dans une première partie, nous nous appuyons sur des résultats de théorie des observateurs non linéaires pour améliorer l'apprentissage des modèles dynamiques. Plus précisément, nous visons à affiner les SSMs tels que ceux obtenus à partir de simulateurs numériques avec des données expérimentales. Pour ce faire, nous nous appuyons sur des résultats théoriques de design d'observateurs non linéaires pour formuler des problèmes de régression adaptés à l'apprentissage d'une représentation spécifique de l'espace d'état à partir des données. Dans une deuxième partie, nous utilisons des techniques d'apprentissage automatique pour permettre le design d'observateurs non linéaires. Nous étudions d'abord comment évaluer l'observabilité des systèmes stochastiques avec des méthodes statistiques, puis comment guider le réglage des observateurs KKL numériques. La thèse est structurée autour des méthodes proposées comme suit :

Le chapitre 3 présente les principaux outils mathématiques nécessaires aux méthodes présentées dans cette thèse. Après une brève introduction sur les processus gaussiens, il fournit les principaux concepts des réseaux de neurones profonds pour l'apprentissage des systèmes dynamiques. Il comprend également un aperçu succinct des méthodes à noyaux nécessaires pour comparer les distributions de sortie des systèmes stochastiques, avant d'exposer le type d'observateur non linéaire le plus utilisé dans cette thèse : l'observateur KKL.

Le chapitre 4 se concentre sur une forme spécifique de SSM, la forme observable canonique avec une non linéarité inconnue. Dans ce cas, les observateurs grand gain garantissent une estimation précise de l'état. Nous interconnectons un tel observateur avec un processus gaussien approximant la non linéarité et montrons que l'état et le modèle dynamique convergent lorsqu'ils sont mis à jour de manière cyclique.

Le chapitre 5 étend cette dernière approche aux systèmes non linéaires généraux avec différents degrés de connaissances préalables, en utilisant la formulation flexible des équations différentielles ordinaires neuronales (NODE). Pour permettre l'apprentissage de NODE à partir d'observations partielles, nous proposons un modèle de reconnaissance qui associe la trajectoire de sortie à l'état latent initial. En particulier, l'exploitation des résultats de la conception des observateurs conduit à un modèle de reconnaissance spécifique adapté aux observateurs KKL.

Le chapitre 6 met l'accent non plus sur l'identification du système mais sur l'estimation de l'état. Plus précisément, nous appliquons des méthodes à noyaux

pour comparer les distributions de mesures provenant de systèmes non linéaires stochastiques et quantifier le degré d'observabilité des états initiaux et des configurations des capteurs à partir de ces données.

Le chapitre 7 étudie ensuite comment l'apprentissage profond peut permettre d'obtenir des observateurs KKL utilisables en pratique en approximant la transformation nécessaire. En particulier, nous proposons de guider la calibration de ces observateurs numériques par le biais d'un critère empirique calculé sur le réseau de neurones modélisant cette transformation.

Pour chacune des méthodes présentées, le code est disponible en open source pour explorer ses capacités et reproduire les résultats :

- Le code correspondant aux Ch. 4–5 : github.com/monabf/obsGP_recogNODE.git;
- Le code correspondant au Ch. 6 : github.com/PFMassiani/data-obs.git;
- Le code correspondant au Ch. 7 : github.com/monabf/learn_observe_KKL.git.

1.3 Publications

Certains des résultats de cette thèse ont été publiés ou soumis pour publication :

Correspondant au Ch. 4 :
[49] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Joint state and dynamics estimation with high-gain observers and Gaussian process models," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1627–1632, 2021

Correspondant au Ch. 5 :
[50] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Recognition Models to Learn Dynamics from Partial Observations with Neural ODEs," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=LTAdaRM29K

Correspondant au Ch. 6 :
[51] P.-F. Massiani, M. Buisson-Fenet, F. Solowjow, F. Di Meglio, and S. Trimpe, "Data-Driven Observability Analysis for Nonlinear Stochastic Systems," *Preprint arXiv:2302.11979*, 2023. (The first two authors contributed equally)

Correspondant auo Ch. 7 :
[52] M. Buisson-Fenet, L. Bahr, and F. Di Meglio, "Towards gain tuning for numerical KKL observers," *Preprint arXiv:2204.00318*, 2022

Chapter 2
Introduction

Numerical simulation is an essential part of industrial development. It enables testing different design choices and evaluating strategies faster and at a lower cost than standard prototyping. Companies such as Ansys have pushed this development forward by producing high-fidelity simulators that precisely predict the behavior of physical systems. Many fields or industries rely on Ansys simulators to examine future projects, from flow-field simulations for the aeronautical industry to models of magnetic motors for electric cars. Even so, numerical simulation faces ongoing challenges, such as increasing the computational speed or improving the models, made imprecise by unmodeled phenomena or parameters that can only be inferred approximately.

The multiplication of data-collecting devices and computing capabilities on physical platforms now gives rise to additional challenges and possibilities: how can the industry leverage the abilities of numerical simulation, while taking advantage of the experimental data its assets generate? Such interrogations lead to the concept of *digital twins* [2]–[5]. Digital twins aim to create an exact numerical replica of a physical object, by combining a high-fidelity simulation model with experimental data gathered on the system. The first step on this path is to construct simulation models running nearly in real-time. Indeed, high-fidelity simulators describing a complex physical process may take several days or weeks to compute, such that incremental updates of the model to match observations can be intractable. Then, the question of merging experimental data into the simulation model can be addressed.

Ansys is working to extend their software suite to include such prospects. To tackle the challenge of computational speed, Ansys is building an in-house reduced order modeling and nonlinear system identification software called DynaROM, which learns a reduced order model (ROM) from training scenarios generated by high-fidelity simulations. In particular, the dimension of the ROM's latent state is inferred by building it iteratively until a desired accuracy is reached, and many other hyperparameters can be chosen automatically. This ROM can then predict complex behaviors such as the fatigue of large gas turbines [6] or the patient-specific evolution of aortic dissections [7] approximately in real-time. Generally, building both accurate and lightweight models is a key challenge for creating digital twins. Many research ideas are emerging to achieve this, e.g., training classifiers to choose between local ROMs [5], [8], [9] or replacing parts of numerical simulators with deep learning models trained from simulation data as in DynaROM [10]. These new research directions motivate this thesis.

The general topic of analyzing experimental data from dynamical systems is of interest for different communities. In this work, we propose to combine structural results stemming from control theory on the one hand, and the data processing and regression power of modern machine learning on the other hand. We present novel methods at this intersection. Data collection on complex systems is costly and not all system variables can be measured. Hence, the main technical challenges lie in the partial, noisy and scarce nature of the observations, paired with an imperfect model. The analysis should thus cover both the dynamics and the state of the system.

We assume that the physical systems at hand can be represented by state-space models (SSMs), as this is a well-established view for which there are many results both in control theory and in machine learning, considering it makes the underlying state explicit. Our goal is then to *develop generic methods for extracting information from experimental data on an underlying state-space model*.

2.1 Related work

To this end, we integrate concepts from control theory and machine learning. In this section, we give a brief overview of the state of the art on this overarching topic, in both communities and at their interface. We distinguish between system identification on the one hand, i.e., determining the dynamics from observations, and state estimation on the other hand, i.e., inferring the latent state. We highlight the gaps in each research direction and insert our contributions into this overview. We mainly focus on works that are relevant for the thesis; a more technical introduction to the most appropriate ones is given in Ch. 3. Directly related work is discussed in more detail in each corresponding chapter.

Machine learning has shown recent success in many static tasks such as image processing or speech generation. However, the search for consistent methods that can tackle the complexity, autocorrelations, and lack of data of the physical world is still open. Hence, there is a growing community at the interface between machine learning and system theory. This community is conducting research on many topics, e.g., reinforcement learning [11], [12] and learning-based control [13], [14]. The subject of system identification is naturally at this interface, as it aims to infer the unknowns of an underlying SSM from measurements. These unknowns can for example be the number of states, parameters of an analytical model, or whole parts of the model. Many methods have been developed in the classic control theory literature to deal with parameter uncertainty. For the linear case, subspace methods aiming to estimate the system matrices are well established [15]. There is no unifying view for nonlinear systems [16], [17], but many approaches from which to choose depending on the use case at hand. These range from optimization-based methods that estimate the unknown parameters of a given model using least squares or related cost functions [18]–[20], Bayesian methods that focus on probabilistic or a posteriori estimates [21], [22], or black-box models that approximate the unknown parts of the model on a suitable basis of functions [23].

However, most of these classic methods are system-specific and require expert knowledge. Hence, recent works leverage modern machine learning to build more general system identification algorithms. For example, Gaussian process state-space models (GPSSMs) [24], [25] have shown promising results thanks to their sample efficiency and analytical formulation, which notably allows for control guarantees [26], [27]. However, learning GPSSMs from partial observations remains an open problem, and most existing approaches rely on heavy nonconvex optimization and do not offer approximation guarantees. This is discussed in Sec. 3.1 following a more technical introduction. We investigate learning such models from partial measurements in Ch. 4.

Another line of work focuses on using deep neural networks to learn dynamics models. For example, sparse identification in a library of physical laws [28], [29], recurrent neural networks (RNNs) [30], [31], or neural ordinary differential equations (NODEs) [32], [33] have been used to identify dynamical systems. These techniques often rely on a transformation from the output to the latent coordinates in which the

dynamics are learned, which can be in the form of an autoencoder [28], [34]. The general question of enforcing physical knowledge in deep learning models has also sparked significant interest [4], [35]–[37]. Nevertheless, approaches that are general but can be adapted to the physical knowledge at hand and can learn from partial observations are still needed. We construct one such approach based on NODEs in Ch. 5. A summary of this literature is provided in Sec. 3.2. Conversely, machine learning methods can also benefit from the lens of control theory. For example, the approximation capabilities of RNNs [38] or NODEs [39] have been investigated via optimal control.

The topic of state estimation is also at the interface between state-space modeling and data analysis, as it aims at inferring the complete latent state of an SSM from partial observations. Hence, there have been attempts to leverage machine learning tools for solving state estimation tasks [40], [41]. In the classical literature on control theory, this is also denoted as observer design. Though there is no established universal observer for nonlinear systems [42], [43], some general designs exist, such as high-gain observers [44], extended Kalman filters (EKF) [45], or Kazantis-Kravaris / Luenberger (KKL) observers [46], [47]. More details are given in Sec. 3.4. These approaches can help extract information from partial observations and precondition the corresponding machine learning problems. Alternatively, modern tools for nonlinear regression can approximate some of the analytically intractable functions present in classic results, e.g., when a transformation to some canonical coordinates is needed, as for high-gain or KKL observers. Hence, exploiting machine learning has the potential to make universal nonlinear observers more applicable in practice. We explore this direction in Ch. 7.

Finally, statistical tools such as kernel methods [48] can also serve to analyze experimental data. They have been used as regression tools for identifying dynamical systems, but can also benefit wider applications by examining more complex objects such as probability distributions. This enables comparing the measurements of stochastic systems and what they reveal about the underlying SSM. For example, the property of observability is nontrivial to define and even more to verify for nonlinear stochastic systems; statistical evaluation of output data enables a step in that direction, as investigated in Ch. 6. A brief overview of these methods is given in Sec. 3.3.

2.2 Outline of the thesis

In this thesis, we *develop generic methods for extracting information from experimental data generated by a dynamical system*. Each method is developed in a specific context, for a specific sub-problem, and corresponds to one of the papers listed below. A technical introduction is provided in Ch. 3, while the main contributions are detailed in Ch. 4–7.

We pursue two lines of work. In a first part, we draw on results from nonlinear observer theory to improve model learning. More specifically, we aim to refine SSMs such as those obtained from numerical simulators with experimental data. For this, we leverage theoretical results from nonlinear observer design to formulate regression problems tailored to learning a specific state-space representation from the given data. In a second part, we employ machine learning techniques to enable data-driven observers. We first investigate how to assess the observability of stochastic systems with statistical methods, then how to guide the tuning of numerical KKL observers. The thesis is structured around the proposed methods as follows:

Chapter 3 presents the main mathematical tools needed for the methods presented in this thesis. After briefly introducing Gaussian process models, it provides the main concepts of deep neural networks for learning dynamical systems. It also includes a succinct overview of kernel methods needed to compare output distributions of stochastic systems, before exposing the type of nonlinear observer most used in this thesis: the KKL observer.

Chapter 4 focuses on a specific form of SSM, the canonical observable form with unknown nonlinearity. In that case, high-gain observers guarantee accurate state estimation. We interconnect the observer with a Gaussian process model for the nonlinearity, and show that both the state and dynamics model converge when updated cyclically.

Chapter 5 extends this latest approach to general nonlinear systems with varying degrees of prior knowledge, making use of the flexible formulation of neural ordinary differential equations (NODEs). To enable learning the NODE from partial observations, a recognition model maps the output trajectory to the initial latent state. In particular, leveraging results from observer design leads to a specific recognition model tailored to KKL observers.

Chapter 6 shifts the focus from system identification to state estimation. More specifically, we apply kernel methods to compare distributions of measurements from stochastic nonlinear systems, and quantify how observable initial states and sensor configurations are given this data.

Chapter 7 then investigates how deep learning can enable practical KKL observers by approximating the necessary transformation. In particular, we propose to guide the tuning of such numerical observers via an empirical criterion computed on the neural network approximator.

For each of the presented methods, open-source code is available to explore its capabilities and reproduce the results:

- Code for Ch. 4–5 is available at: github.com/monabf/obsGP_recogNODE.git;
- Code for Ch. 6 is available at: github.com/PFMassiani/data-obs.git;
- Code for Ch. 7 is available at: github.com/monabf/learn_observe_KKL.git.

2.3 Publications

Some of the results of this thesis have been published or submitted for publication:

Corresponding to Ch. 4:
[49] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Joint state and dynamics estimation with high-gain observers and Gaussian process models," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1627–1632, 2021

Corresponding to Ch. 5:
[50] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Recognition Models to Learn Dynamics from Partial Observations with Neural ODEs," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=LTAdaRM29K

Corresponding to Ch. 6:

[51] P.-F. Massiani, M. Buisson-Fenet, F. Solowjow, F. Di Meglio, and S. Trimpe, "Data-Driven Observability Analysis for Nonlinear Stochastic Systems," *Preprint arXiv:2302.11979*, 2023. (The first two authors contributed equally)

Corresponding to Ch. 7:
[52] M. Buisson-Fenet, L. Bahr, and F. Di Meglio, "Towards gain tuning for numerical KKL observers," *Preprint arXiv:2204.00318*, 2022

Chapter 3
Mathematical tools

Résumé Ce chapitre présente les principaux concepts et outils mathématiques utilisés dans cette thèse. En premier lieu, certaines méthodes de modélisation de systèmes dynamiques à partir de données sont brièvement introduites. Nous commençons par les processus gaussiens, en particulier pour les modèles sous forme d'état, et leur utilisation pour l'apprentissage de dynamiques dans la Sec. 3.1. Nous discutons ensuite de l'utilisation des modèles d'apprentissage profond pour l'identification des systèmes dynamiques dans la Sec. 3.2. Nous examinons ensuite certaines approches d'apprentissage profond particulièrement bien adaptées aux systèmes dynamiques : les équations différentielles ordinaires neuronales et les réseaux de neurones récurrents. Par la suite, l'accent est mis sur l'estimation d'état à partir de données de mesure. Pour cela, nous présentons succinctement les méthodes à noyaux utilisées pour analyser des données statistiques dans la Sec. 3.3, en particulier un test statistique pour distinguer deux distributions à partir d'échantillons. Enfin, nous présentons un observateur d'état non linéaire particulier dans la Sec. 3.4 : l'observateur de Katzantis-Kravaris/Luenberger, dont nous rappelons les principaux résultats.

Abstract This chapter presents the main mathematical concepts and tools used in this thesis. First, some methods for modeling dynamical systems from data are briefly introduced. We start with Gaussian processes, and in particular Gaussian process state-space models and their use for dynamics learning in Sec. 3.1. We then discuss the use of deep learning models for identifying dynamical systems in Sec. 3.2. Then, the focus shifts to state estimation from measurement data. For this, we succinctly introduce kernel methods used to analyze statistical data in Sec. 3.3, in particular a kernel two-sample test to distinguish distributions based on samples. Finally, we present a specific nonlinear state observer in Sec. 3.4: the Katzantis-Kravaris/Luenberger observer, for which we recall the main results.

Notations In this thesis, all vectors are column vectors by convention, and the norm considered is the Euclidean ℓ_2 norm for vectors and matrices denoted by $|\cdot|$, unless stated otherwise.

3.1 Learning dynamics with Gaussian processes

Gaussian processes (GPs) are a powerful machine learning framework, with good sample efficiency, providing probabilistic estimates. In Ch. 4, we focus on their use for regression problems. GPs have gained interest over the last years; an extensive overview is provided in [53]. In this section, we briefly introduce their mathematical formulation, then discuss their recent application to system identification, and the main trends to overcome their initial shortcomings in this context.

3.1.1 Preliminaries on Gaussian processes

Let us recall the main definitions and interpretations of GPs as a tool for statistical regression; see [53] for an overview.

Definition 1 ([53, Sec. 2.2]). *A Gaussian process (GP) is a collection of random variables, any finite subset of which is jointly normally distributed. A GP can be indexed over a set $\mathcal{X}$. In other words, if f is a GP, it associates each subset $X \in \mathcal{X}$ with a collection of jointly normally distributed random variables $f(X) = (f(x_i))_{x_i \in X}$. It is fully characterized by its mean function $m(\cdot)$ and its covariance function $k(\cdot, \cdot)$.*

For simplicity, we assume $m \equiv 0$; see [53, Sec. 2.7] for a discussion. Let $\mathcal{X} = \mathbb{R}^{nd_x}$ and consider a finite subset $X \in \mathcal{X}$:

$$X = \begin{pmatrix} x_0 & \dots & x_n \end{pmatrix} \tag{3.1}$$

where each $x_i \in \mathbb{R}^{d_x}$, and a set of measured outputs

$$Y = \begin{pmatrix} y_0 & \dots & y_n \end{pmatrix} \tag{3.2}$$

such that

$$y_i = f(x_i) + \epsilon_i, \tag{3.3}$$

where each $y_i \in \mathbb{R}$ and each ϵ_i is a realization of Gaussian measurement noise of variance σ_ϵ^2.

Definition 1 implies a consistency requirement, i.e., that any collection of values $f(x)$ be consistently normally distributed. Hence, the prediction of f at a new, unobserved point x can be obtained by conditioning over the inputs X and outputs Y knowing σ_ϵ^2. More precisely, it is normally distributed with posterior mean and variance, respectively given by

$$\mu(x|X,Y) = \underline{k}(x)^\top (K + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} Y \tag{3.4}$$

$$\sigma^2(x|X,Y) = k(x,x) - \underline{k}(x)^\top (K + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} \underline{k}(x) \tag{3.5}$$

where $\mathbb{I}_n$ is the identity matrix of size n, k is the kernel function, $K = (k(x_i, x_j))_{x_i, x_j \in X}$ is the covariance matrix, and $\underline{k}(x) = (k(x_i, x))_{x_i \in X}$. This straightforwardly extends to multiple test points by considering their covariance matrix instead of the variance at one point. By a slight abuse of notation, for any fixed x, the probability of drawing the associated value $f(x)$ conditioned on the data follows a normal distribution $\mathcal{N}\left(\mu(x), \sigma^2(x)\right)$. This is denoted as the posterior distribution of f at x, and it is the formula we will mostly use to apply GP regression in practice.

If the output is of dimension $d_y > 1$, one can assume all output dimensions are independent and select one kernel function for each.

Definition 2. *In the case of d_y independent output dimensions with d_y different kernels, we obtain a posterior covariance matrix of the form*

$$\Sigma(x) = \begin{pmatrix} \sigma_1^2(x) & & 0 \\ & \ddots & \\ 0 & & \sigma_{d_y}^2(x) \end{pmatrix}, \tag{3.6}$$

where $\sigma_i^2(x)$ is the posterior variance for the i^{th} output dimension as per definition (3.5), using the i^{th} kernel. We then define the scalar posterior variance in the multioutput case as

$$\sigma^2(x) = |\Sigma(x)|^{1/d_y}. \tag{3.7}$$

In this thesis, we always consider the scalar posterior variance when making predictions, be it (3.5) for a single output dimension or (3.7) for multiple independent output dimensions. Otherwise, one can formulate the posterior distribution for multioutput GPs as in [54] so that the output dimensions correlate, at the cost of significant computational overhead.

Function properties such as smoothness or periodicity can be encoded in the choice of the kernel $k(\cdot,\cdot)$, which acts as a similarity measure for the values of $f(\cdot)$. This kernel usually depends on some so-called hyperparameters, often obtained by maximizing the data marginal log-likelihood

$$\log p(Y|X) = -\frac{1}{2}Y^\top (K + \sigma_\epsilon^2 \mathbb{I}_{d_x})^{-1} - \frac{1}{2}\log|K + \sigma_\epsilon^2 \mathbb{I}_n x| - \frac{n}{2}\log 2\pi, \tag{3.8}$$

with a gradient-based optimization method and multiple starts. In this thesis, we mainly use the anisotropic squared exponential kernel

$$k(x, x') = s^2 \exp\left(-\sum_{i=1}^{d_x} \frac{(x_i - x_i')^2}{2l_i^2}\right) \tag{3.9}$$

where the hyperparameters are $l = (l_1, ..., l_{d_x})^\top$ the vector of lengthscales and s^2 the scaling factor. For so-called universal kernels such as the squared exponential, given any choice of hyperparameters, the GP posterior mean can approximate any sufficiently smooth function arbitrarily well as the amount of data grows to infinity, i.e., it has a property of universal approximation [53], [55]. However, for a fixed amount of data, learning performance can significantly degrade if the kernel or its parameters are badly chosen, even though overfitting tends to be less of a problem for GP regression than for most other machine learning frameworks [26], [56].

Sparse approximations

GP models tend to be computationally heavy as adding a new data point requires the inversion of the large covariance matrix K, for which the temporal complexity is $O(n^3)$. Therefore, constructing scalable GP models that maintain performance while speeding up computations is an active area of research. This is particularly relevant for performing system identification with GPs, as the amount of data grows over time.

A first approach is to design sparse approximations of GPs. An overview of the most important methods is provided in [53, Ch. 8], and more recently in [57]. Most sparse approximations rely on a vector of inducing variables, which are values of the GP at artificially chosen inducing inputs. It is then assumed that the function values at the test points are conditionally independent of those at the training inputs given these inducing variables, and the posterior distribution conditioned on the inducing variables is approximated in a computationally efficient fashion. The inducing variables can be a subset of the data, or optimized jointly with the hyperparameters.

Another line of work consists in directly approximating the GP posterior distribution: this method is denoted variational inference [58]–[60]. Variational inference approximates a conditional distribution $p(z|x)$, where z are the latent variables while

x are the observations, with a simpler distribution $q(z)$, such as a mixture of Gaussians, whose parameters are optimized to best fit the true distribution. Here, the latent variables z often play the same role as the inducing variables for classical sparse GPs, and are jointly optimized with the natural parameters of q. All variational parameters are chosen by minimizing the Kullback-Leibler divergence between the exact and approximate posterior, which boils down to maximizing the evidence lower bound (ELBO):

$$\text{ELBO}(q) = E\left[\log p(x,z)\right] - E\left[\log q(z)\right], \tag{3.10}$$

where $q(z)$ is known and $p(x,z)$ can be computed explicitly in the case of GPs, or through approximate inference. The ELBO is generally nonconvex, but this procedure can take advantage of recent tools such as stochastic or distributed optimization to find a suitable local optimum, and results in an approximate posterior which is easy to sample, yielding fast predictions.

Finally, some other works focus on recursive approximations to speed up training, e.g., [61].

3.1.2 Gaussian process state-space models

GPs have become a popular tool for learning dynamical systems [24], [62], with applications among others in robotics [14], safe learning or exploration [26], [27], [63], or reinforcement learning [64], [65]. They provide an analytical formulation, which is suitable for further analysis, in particular for giving probabilistic guarantees [66], [67] or for uncertainty-based active learning [68], [69]. They are also data-efficient [25] and provide an automatic trade-off between model smoothness and data fitness through the Bayesian approach [69].

The modeling problem is often formulated as a Gaussian process state-space model (GPSSM) [70], i.e., a discrete-time state-space model of form

$$\begin{aligned} x_{t+1} &= f(x_t) + \eta \\ y_t &= h(x_t) + \epsilon \end{aligned} \tag{3.11}$$

where $x \in \mathbb{R}^{d_x}$ is the state of the system, $y \in \mathbb{R}^{d_y}$ is the measured output, and η resp. ϵ are Gaussian process resp. measurement noise. The transition or dynamics function f and the measurement function h are assumed to be GPs, usually with independent output dimensions.

Collecting observations of (3.11) naturally leads to a dataset that grows over time; sparse approximations with a fixed number of inducing variables can alleviate this issue. In any case, sampling from GPSSMs is highly nontrivial. Indeed, for a given state x_0, the next state x_1 follows a normal distribution as per (3.11), but not necessarily the following states: if f is simply deterministic but nonlinear, it deforms the normal distribution. Hence, exact sampling can quickly get computationally intractable, so that approximations have been investigated. For example, one can linearize the GP posterior to maintain Gaussian distributions over several time steps, or use moment matching to approximate the posterior at each time step by a Gaussian, whose mean and variance are computed with exact inference [25, Sec. 4]. However, this is still computationally expensive and can degrade performance. Recent research thus proposes more sophisticated approximations [71]. Since sampling from the posterior is already complex, h is often assumed to be a linear, deterministic function for simplicity. Other than these, several more practical interpretations of GPSSMs have been suggested

to circumvent the sampling difficulty [26, Sec. 2.3]. The *deterministic* view only uses the mean of the GP posterior distribution as a function approximator, disregarding the probabilistic aspects. The *stochastic* view draws independent samples at each time step, while the *belief-space* interpretation considers a deterministic transition function for the mean on the one hand, and the variance on the other hand. In Ch. 4 of this thesis, we follow the deterministic view: we use the mean of the GP posterior distribution as a function approximator to estimate the true dynamics, without considering the probabilistic aspects. For the graphical representations, we simply show the variance at the given deterministic point for illustration purposes, but only consider the mean in all computations and interpretations.

Physical priors in GPSSMs

One of the advantages of GPSSMs is that many forms of prior knowledge about the physical system can be included. For example, if the function generating the data is known to be periodic, this can be encoded via a periodic kernel of form

$$k(x, x') = s^2 \exp\left(-\sum_{i=1}^{d_x} \frac{2\sin^2(\pi|x - x'|/p)}{l_i^2} \right) \tag{3.12}$$

where s is a scaling parameter, $l = (l_1, ..., l_{d_x})^\top$ is the vector of lengthscales and p is the period.

Some structural constraints directly lead to a modified posterior distribution, so that the resulting samples necessarily satisfy them, e.g., mechanical force constraints [72]–[74] or linear ordinary differential equations (ODEs) with constant coefficients [75]. Other properties such as conservation laws can be enforced by suitable numerical integrators in specific cases, such as symplectic integrators for Hamiltonian systems; embedding a chosen integrator into a variational approximation of the GP posterior [76] or combining exact GP inference with variational integrators [77] also leads to structure-preserving predictions.

Stability constraints can also be imposed, e.g., using control Lyapunov functions [26], [78]. For example, if the equilibrium points of the system are known, or their number is known and they are inferred from data, they can be enforced in the GP model f by treating them as virtual inputs $\underline{x}$ such that $f(\underline{x}) = \underline{x}$. This yields an updated posterior distribution. Then, a virtual control u is added to the GP such that the prediction at a test point x_t is $f(x_t) + u(x_t)$. This control law $u(\cdot)$ is obtained by solving a convex optimization problem, ensuring that all equilibrium points are asymptotically stable for a given convex Lyapunov functional, while minimally perturbing the GP model. This yields an accurate and stable GPSSM [78].

Finally, it is also possible to model the residuals of a known dynamics function with GPSSMs [79], i.e., learn the difference between the ground truth and a prior model, by fitting the GP to observations from which the prior predictions have been subtracted, or, equivalently, by using the known function as m in the GP posterior mean (3.4).

Joint inference and learning for GPSSMs

Many works on GPSSMs assume full state observations are available. However, if only partial observations y can be collected, identifying the GPSSM becomes much more complex. Indeed, not only do f and h need to be inferred from data (learning problem), but the posterior distribution of the unknown latent state x given the

observations is also needed (inference problem): this is denoted as joint inference and learning.

One of the most widespread takes on this problem is variational inference [58]. As seen in Sect. 3.1.1, variational inference consists in approximating a complex conditional distribution with a simpler distribution q, then optimizing the parameters of q to minimize the ELBO. This approach can be used not only to speed up predictions as in [59], but also to perform joint inference and learning in GPSSMs [80]. In this case, the GPSSM is augmented with inducing variables u which are jointly normally distributed with the latent function values. The approximating distribution q is chosen such that it factorizes similarly to the joint distribution of a GPSSM, and the ELBO is computed explicitly. Then, stochastic gradient descent is performed to optimize all hyperparameters and variational parameters, which describe the distribution of the inducing variables. This approach yields a tractable posterior distribution even for a latent space that is larger than the observation space. It has since been expanded to improve prediction accuracy by imposing a more structured variational distribution, parametrized by a bi-directional recognition model [81], taking into account temporal correlations between the latent states [82], treating the temporal states jointly through the Laplace approximation [83], or running a backward pass during inference to take several observations into account and predict even unstable systems [84]. A closed-form, full analytical variational inference scheme is also presented in [85].

Another line of work applies the expectation-maximization algorithm [70]. After introducing a set of inducing variables also denoted pseudo training set, the algorithm alternates between inference steps, in which the posterior distribution is approximated given all current hyperparameters and inducing variables, and learning steps, in which these parameters are updated to maximize a lower bound on the log-likelihood of the training data given the parameters.

3.2 Deep learning for dynamical systems

Deep learning utilizes deep artificial neural networks to solve tasks based on data. Many textbooks and resources are available to understand these notions and apply them in practice, e.g., [86]–[88]. Different types of architectures of neural networks exist in the literature and serve different purposes: in image processing, convolutional neural networks apply a moving filter over the input image to extract a feature map that captures spatial correlations; in time series and natural language processing, long short-term memory networks (LSTMs) have been successful by retaining both a long-term cell state and a short-term hidden state that are combined with the current input via a series of gates; in data compression, autoencoders learn a projection of the input into a lower-dimensional space and back to the original space... In this section, we merely graze the surface of deep learning by shortly introducing the tools that are most relevant for this thesis. We start with a technical introduction to one of the most basic types of neural networks: multilayer perceptrons, which we will mainly use in the following chapters. We then succinctly describe a few other architectures that are meaningful for the topics at hand.

3.2.1 Introduction to feed-forward neural networks

An artificial neural network (NN) is a structure that takes an input x and returns an output $f_\theta(x)$, where θ are the parameters of the network. The output of a single layer is given by $a(w^\top x + b)$, where x is the input to the layer, w is a vector of weights, b a

vector of biases, and a is a (generally nonlinear) activation function. For feed-forward networks, the output of each layer serves as input to the next, and all weights and biases are concatenated into one vector of parameters θ. More complex architectures can also include loops or cycles to keep information inside the network.

In supervised learning, the data is labeled: for each input x, there is a known output y. The parameters of the NN are then adapted for the estimated output $f_\theta(x)$ to match the true output y on a training set of N points as closely as possible, usually by minimizing an empirical risk in the form of a least squares loss:

$$L(\theta) = \sum_{i=1}^{N} |y_i - f_\theta(x_i)|^2. \tag{3.13}$$

This is denoted as *training* the network. This optimization is typically conducted by iteratively updating the parameters following a stochastic gradient descent scheme. At each step, such methods compute the average gradient on a random subset of training examples called minibatch, until a convergence criterion is met. They have been further improved, e.g., by accelerating the descent with momentum [89] or using an adaptive step size. In this thesis, we mainly use Adam [90], one of the most prominent algorithms to date.

To minimize the loss L with gradient descent methods, it is necessary to compute the gradient of L w.r.t. the network parameters θ. This is at the core of NN training, and is achieved with *backpropagation*. This procedure relies on the chain rule: since the output of the NN is obtained by applying a series of simple operations to the input layer by layer, one can compute the gradient of the output w.r.t. each parameter by going backwards through all the layers using the chain rule [88, Ch. 1], [87, Sect. 4.6]. This can be computed analytically for a fixed NN. However, most modern machine learning libraries have tools for automatic differentiation [91]: when describing a series of operations such as the layers of a NN, a computational graph is built that keeps track of the dependencies and intermediate derivatives. This is the forward pass. Then, the backward pass is called given a scalar value for the loss: it goes through all operations again and combines the necessary partial derivatives using the chain rule. Stochastic gradient descent can then be applied with the current gradient computation for the current minibatch. Note that efficient automatic differentiation is one of the main ingredients that has enabled the success of deep learning, and it can now be used in many other fields thanks to optimized general-purpose libraries such as `JAX`. In this thesis, we use `PyTorch`, which enables automatic differentiation through the `autograd` package and the `jacrev` package for more complex operations.

Many practical aspects should be considered for successfully training a NN on a supervised learning task. For example, overfitting remains one of the main issues: it is possible to minimize the empirical risk by exactly reproducing all training examples, but this leads to a model that is incapable of generalizing, i.e., correctly predicting examples outside of the training set. Tricks such as regularization (adding terms to the loss that penalize a norm of θ), early stopping (monitoring the loss on a separate validation set and stopping when it starts to rise) or cross-validation (training several models on different subsets of training and validation data) help alleviate this issue. Such practical considerations are necessary to obtain accurate NN models; an overview and best practices are presented in [88], [92]. They are not further discussed in this thesis, except for relevant aspects that have influenced some concrete examples.

This brief introduction describes feed-forward NNs, which include multilayer perceptrons, i.e., vanilla NNs, simply denoted as NNs in the following chapters,

but also convolutional NNs. In this thesis, we mainly use multilayer perceptrons as function approximators in a supervised context, and a few other types of architectures concisely introduced in the next section.

3.2.2 Neural networks for physical systems

Great effort has recently been made to bridge the gap between the latest advances in deep learning and more established results or concepts in the physical sciences. Overviews of the works on integrating deep learning into studies of physical systems are given in [4], [35]–[37]. We briefly introduce the types of models that are most relevant for this thesis in the next sections. Note that for many of these works, the coordinate system in which physical knowledge can be included is not assumed to be the observed coordinates. Hence, a change of variables on the state of the underlying SSM is often learned jointly, commonly in the form of an autoencoder [29], [93].

Physics-informed neural networks

In the seminal paper [94], solutions of known partial differential equations (PDEs) are approximated with deep NNs. This approach is denoted as physics-informed neural networks (PINNs). The solution $u(t, x)$ is approximated by a NN whose parameters are optimized to minimize the least squares loss $L_u + L_f$, where L_u penalizes the error on a dataset of trajectories, while L_f computes the differential operator applied to the NN model on a spatio-temporal grid using automatic differentiation, and penalizes the residuals of that differential equation. The addition of the unsupervised penalty L_f is expected to have regularization properties and enable some level of extrapolation, usually a shortcoming of NNs. The method is demonstrated on both continuous and discrete-time systems and can capture the nonlinear behavior of the solution even with a low number of data points if the dataset is informative enough. It is also possible to optimize some unknown parameters of the PDE jointly with the solution.

PINNs have since been used for various applications and combined with several other concepts and frameworks. For example, in [95], PINNs are combined with Lyapunov stability theory to learn regions of attraction of autonomous dynamical systems; in [96], they are merged with autoencoders to discover internal variables and adapted to integrate the laws of thermodynamics and describe the behavior of complex inelastic materials. A related method, which retrospectively falls into the same category, is the line of work on Hamiltonian and Lagrangian networks originating from [97]–[99] (see [100] for an overview). The system is assumed to follow Hamiltonian (resp. Lagrangian) dynamics, and the Hamiltonian (resp. Lagrangian) function is approximated with a NN by penalizing the residuals of the corresponding PDE on a set of training points. The resulting dynamics are then used to predict the evolution of the system. This approach has also led to many applications and extensions. For example, in [101], standard Hamiltonian NNs are extended to learn dissipative systems, whose dynamics are the addition of a conservative and a dissipative field, approximated jointly by two NNs.

Recurrent neural networks

Recurrent neural networks (RNNs) are structurally different from feed-forward networks [87, Ch. 7]. While the latter are designed for independent input points, RNNs are built for sequential inputs. They capture temporal correlations by storing the

information contained in previous time steps in a so-called memory, in the form of a hidden state that is passed on through a feedback loop in the recurrent layers. Backpropagation is then used to train the network by unfolding this feedback loop, i.e., tracing back the sequence of operations necessary to obtain the output for each time step in the input sequence. There are different architectures of RNNs used for various applications. In particular, the hidden state and input are combined through so-called gates, whose parameters are also optimized during training. LSTMs [102], [87, Sect. 7.5] are a form of RNNs that is widely used in natural language processing, where a hidden and a cell state keep track of a short-term and a long-term context. At each time step, these two states are updated and combined with the new information from the current input through a forget gate, an input gate and an output gate to produce the output.

In this thesis, we shortly make use of gated recurrent units (GRUs), first introduced in [103]. In this type of RNN, one hidden state is kept in the recurrent layer through a feedback loop. At each time step in the input sequence, this hidden state is updated via a combination of update and reset gates, then the output is predicted. GRUs have been successfully applied for time series forecasting and other sequential data processing tasks.

Neural ordinary differential equations

Neural ordinary differential equations (NODEs) were first proposed in [32] and have since gained significant interest. They aim to learn a vector field that generates the data with a feed-forward NN; see [33] for an overview. Contrarily to PINNs, we are not approximating the solution of a known ODE (or PDE), but rather the vector field that characterizes this ODE using known solutions, i.e., trajectories. The network is trained by minimizing the difference between the trajectory resulting from the current model and the true trajectory, usually by automatically differentiating through the numerical solver used for simulating the current estimate. NODEs can be interpreted as continuous-depth neural networks, as the input evolves continuously during the forward pass. These models have found various applications, from image classification [33] or analysis of complex materials [104] to generative models for climate forecasting [105]. Their formulation is particularly natural for modeling dynamical systems with different degrees of prior knowledge. We introduce NODEs more formally, go deeper into the related work and investigate this last direction in Ch. 5.

3.3 Kernel methods

Kernel methods are an active research area related to statistical learning theory [48]. They form a modular framework that can efficiently solve many machine learning tasks. In the following sections, we give the minimal intuitive treatment necessary to introduce the tools used in Ch. 6 of this thesis.

3.3.1 Reproducing kernel Hilbert spaces

Kernel methods rely on the theory of reproducing kernel Hilbert spaces (RKHSs). An intuitive introduction can be found in [106]; see [48], [107] for more details.

Kernel methods can be motivated as follows [106]. Many classical learning algorithms mainly compute inner products between data points. These can be interpreted

as similarity measures between samples. Kernel methods replace these inner products, which may be too restrictive for the learning problem at hand, with more flexible, possibly nonlinear similarity measures. This is achieved by embedding the input x into a feature $\Phi(x)$, possibly high-dimensional, then computing the similarity measures directly in the feature space. In the context of kernel methods, the feature space is an RKHS, i.e., a Hilbert space over functions in which the evaluation functionals are bounded [106, Def. 2.4]. It can be shown that RKHSs have the point evaluation property: for any input x, there is a representer function in the RKHS that expresses the evaluation $f(x)$ as an inner product, for any f in the RKHS; see [106, Sec. 2.2] for details. We now introduce these tools more formally.

Let $\mathcal{X}$ be a set. A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be positive definite if it satisfies the symmetry property

$$k(x, y) = k(y, x) \qquad \forall\, x, y \in \mathcal{X} \tag{3.14}$$

and the positivity property

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geq 0 \qquad \forall\, c_i \in \mathbb{R}, x_i \in \mathcal{X}, i \in \{1, \ldots, N\}. \tag{3.15}$$

This corresponds to its Gram matrix $(k(x_i, x_j))_{i,j \in \{1,\ldots,N\}}$ being positive semi-definite. It is shown in [108] that for any fixed, positive definite kernel k, there exists a unique corresponding RKHS denoted by $\mathcal{H}_k$. This RKHS [109, Def. 2.3] is a Hilbert space over the functions from $\mathcal{X}$ to $\mathbb{R}$, equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$, for which the *canonical feature map*

$$k_x : y \in \mathcal{X} \mapsto k(x, y) \tag{3.16}$$

belongs to $\mathcal{H}_k$ and has the *reproducing property*

$$f(x) = \langle f, k_x \rangle_{\mathcal{H}_k} \qquad \forall\, f \in \mathcal{H}_k, x \in \mathcal{X}. \tag{3.17}$$

Hence, k is called a *reproducing kernel*. Thus, the feature map k_x represents point evaluation [106, Prop. 2.1], i.e., any function $f \in \mathcal{H}_k$ can be represented with the reproducing kernel as

$$f(x) = \sum_{i=1}^{\infty} c_i k_{x_i}(x) = \sum_{i=1}^{\infty} c_i k(x_i, x) = \langle f, k_x \rangle_{\mathcal{H}_k} \qquad \forall\, x \in \mathcal{X}, \tag{3.18}$$

with coefficients $c_i \in \mathbb{R}$. The reproducing kernel can also be interpreted as a similarity measure between inputs, such that the feature map acts as a similarity measure between x and all other points in $\mathcal{X}$. In particular, we have the so-called *kernel trick*:

$$k(x, y) = \langle k_x, k_y \rangle_{\mathcal{H}_k} \qquad \forall\, x, y \in \mathcal{X}, \tag{3.19}$$

which enables easy computations of scalar products of finite linear combinations of the reproducing kernel k_x. Hence, instead of working in the unstructured set $\mathcal{X}$, one can work in the structured Hilbert space $\mathcal{H}_k$, in which an inner product and the corresponding norm are defined, and in which the kernel trick enables fast computations.

This view is widely used in statistical learning theory, specifically due to the generalized representer theorem [110]. Roughly speaking, this representer theorem

states that given a finite number of training points x_i and corresponding observations y_i, $i \in \{1, \dots, N\}$, concatenated in X, Y, any solution of the regularized empirical risk minimization problem

$$\min_{f \in \mathcal{H}_k} E(X, Y, f) + R(|f|_{\mathcal{H}_k}) \tag{3.20}$$

where E is an error functional depending on x_i, y_i and the estimated values $f(x_i)$, $i \in \{1, \dots, N\}$, and R is a strictly increasing real-valued regularization term, is of the form

$$f^*(\cdot) = \sum_{i=1}^{N} \alpha_i k(x_i, \cdot) \tag{3.21}$$

with $\alpha_i \in \mathbb{R}$, $i \in \{1, \dots, N\}$ some coefficients. Hence, any solution of the regularized empirical risk minimization problem in $\mathcal{H}_k$ can be written as a linear combination of the kernel values at the training points. In particular, the kernel Ridge regression

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{N} |y_i - f(x_i)|^2 + \lambda |f|^2_{\mathcal{H}_k} \tag{3.22}$$

with $\lambda > 0$ falls into this category. The solution is necessarily of form (3.21), the vector α^* of coefficients

$$\alpha^* = (K + \lambda \mathbb{I})^{-1} Y, \tag{3.23}$$

where $K_{ij} = k(x_i, x_j)$, $i, j \in \{1, \dots, N\}$ being a possible solution (unique if K is invertible). This is obtained by replacing the form of f^* into the cost function and setting its gradient to be zero at the optimum [109, Th. 3.4].

To conclude, choosing a positive definite kernel k amounts to choosing a similarity measure between the points x, so that in the corresponding RKHS denoted $\mathcal{H}_k$, every function can be represented based on this similarity measure. Nonlinear regression problems then lead to solutions in the form of linear combinations of the similarity measure evaluated at the training points. Casting certain problems into an RKHS enables manipulating complex objects in a structured space, and reduces them to a finite number of evaluations of the kernel function.

Note that Gaussian processes presented in Sect. 3.1 are related to kernel methods. In particular, the solution of the kernel Ridge regression problem of form (3.21) with α^* above is the posterior mean of a GP model with $\sigma_\epsilon^2 = \lambda$. However, interpreting the covariance in this context remains a topic of investigation; see [53, Ch. 6][109] for more details.

3.3.2 Statistical testing with kernels

Kernel machines embed a point $x \in \mathcal{X}$ into a feature space through a so-called feature map k_x, corresponding to the chosen similarity measure. However, $\mathcal{X}$ needs not be a set of points: a reproducing kernel and the corresponding RKHS can be defined over more complex objects, e.g., graphs, strings or images. In particular, probability distributions can be represented as objects in an RKHS through kernel mean embeddings [106]. Let $\mathbb{P}$ be a probability distribution on $\mathcal{X}$; its kernel mean

embedding (KME) denoted $\mu_{\mathbb{P}}$ is given by

$$\begin{aligned} \mu_{\mathbb{P}} : \mathbb{P} &\to \mathbb{R} \\ x &\mapsto \int_{\mathcal{X}} k(x, x') \mathrm{d}\mathbb{P}(x') = E_{X \sim \mathbb{P}} \left[k(x, X) \right]. \end{aligned} \tag{3.24}$$

This is the average value of $k(\cdot, X)$, where $X \sim \mathbb{P}$. One can then compare such KMEs in the RKHS. For example, the maximum mean discrepancy (MMD) between two distributions $\mathbb{P}$ and $\mathbb{Q}$ is defined as the distance between their embeddings in the RKHS:

$$\mathrm{MMD}[\mathbb{P}, \mathbb{Q}] = |\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}|_{\mathcal{H}_k}. \tag{3.25}$$

These KMEs pave the way to manipulating probability distributions in a structured space. Though the theoretical values of the KME and MMD often cannot be computed analytically, they can be estimated empirically from independent and identically distributed (i.i.d.) samples from the source distributions $\mathbb{P}$ and $\mathbb{Q}$ [106]. This enables constructing a kernel two-sample test [111] to compare distributions based on samples. The statistic in this test is the empirical estimate of the MMD. Given concentration bounds on this estimate, a threshold can be determined depending on the number of samples, such that if the empirical MMD is above this threshold, then $\mathbb{P}$ and $\mathbb{Q}$ are different with high probability. Such lines of work comparing distributions based on their KMEs have since led to many applications, and efforts are made to extend them to stochastic dynamical systems, e.g., to determine whether two sets of full-state measurements stem from the same stochastic system [112]. In Ch. 6, we make use of these concepts to develop a data-based observability test for nonlinear stochastic systems.

3.4 The Katzantis-Kravaris/Luenberger observer

Nonlinear observer design is a broad and active research field, whose aim is to estimate the state of nonlinear systems from partial measurements. There is no established design with convergence guarantees for general nonlinear systems, as one of the most widely used observers is the extended Kalman filter (EKF) [45], which only provides local convergence guarantees. The Katzantis-Kravaris/Luenberger (KKL) observer, also denoted nonlinear Luenberger observer, provides global convergence guarantees and has recently gained interest. However, many questions remain open to make it applicable in practice. In this thesis, we use the KKL observer as a general state estimator and investigate how to tune it. We briefly recall the results on this subject that are most relevant for this in the following sections.

3.4.1 Autonomous KKL observers

We first recall the main existence results of KKL observers for autonomous systems. Consider the following autonomous nonlinear dynamical system

$$\begin{aligned} \dot{x} &= f(x), \\ y &= h(x) \end{aligned} \tag{3.26}$$

where $x \in \mathbb{R}^{d_x}$ is the state, $y \in \mathbb{R}^{d_y}$ is the measured output, f is a C^1 function and h is a continuous function. The goal of observer design is to compute an estimate of the

state $x(t)$ using the past values of the output $y(s)$, $0 \leq s \leq t$. We make the following assumptions:

Assumption 1. *There exists a compact set $\mathcal{X}$ such that for any solution x of (3.26), $x(t) \in \mathcal{X}$ $\forall\, t \geq 0$.*

Assumption 2. *There exists an open bounded set $\mathcal{O}$ containing $\mathcal{X}$ such that (3.26) is backward $\mathcal{O}$-distinguishable on $\mathcal{X}$, i.e., for any trajectories x_a and x_b of (3.26), there exists $\bar{t} > 0$ such that for any $t \geq \bar{t}$ such that $(x_a(t), x_b(t)) \in \mathcal{X} \times \mathcal{X}$ and $x_a(t) \neq x_b(t)$, there exists $s \in [t - \bar{t}, t]$ such that*

$$h(x_a(s)) \neq h(x_b(s))$$

and $(x_a(\tau), x_b(\tau)) \in \mathcal{O} \times \mathcal{O}$ for all $\tau \in [s, t]$. In other words, their respective outputs become different in backward finite time before leaving $\mathcal{O}$.

This assumption is denoted *backward distinguishability*. It means that the current state is uniquely determined by the past values of the output. On the contrary, (forward) distinguishability means that the initial state is uniquely determined by the future values of the output. If the solutions of (3.26) are unique, e.g., if f is C^1, these two notions are equivalent.

The following Theorem derived in Andrieu and Praly [47] proves the existence of a KKL observer.

Theorem 1 (Andrieu and Praly [47])**.** *Suppose Assumptions 1 and 2 hold. Define $d_z = d_y(d_x + 1)$. Then, there exists $\ell > 0$ and a set S of zero measure in $\mathbb{C}^{d_z}$ such that for any matrix $D \in \mathbb{R}^{d_z \times d_z}$ with eigenvalues $(\lambda_1, \ldots, \lambda_{d_z})$ in $\mathbb{C}^{d_z} \setminus S$ with $\Re\lambda_i < -\ell$, $i \in \{1, \ldots, d_z\}$, and any $F \in \mathbb{R}^{d_z \times d_x}$ such that (D, F) is controllable, there exists an injective mapping $\mathcal{T} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ that satisfies the following equation on $\mathcal{X}$*

$$\frac{\partial \mathcal{T}}{\partial x}(x) f(x) = D\mathcal{T}(x) + Fh(x), \tag{3.27}$$

and its left inverse $\mathcal{T}^* : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ such that the trajectories of (3.26) remaining in $\mathcal{X}$ and any trajectory of

$$\dot{z} = Dz + Fy \tag{3.28}$$

satisfy

$$|z(t) - \mathcal{T}(x(t))| \leq M|z(0) - \mathcal{T}(x(0))|e^{-\lambda_{\min} t} \tag{3.29}$$

for some $M > 0$ and with

$$\lambda_{\min} = \min\left\{|\Re\lambda_1|, \ldots, |\Re\lambda_{d_z}|\right\}. \tag{3.30}$$

This yields

$$\lim_{t \to +\infty} |x(t) - \mathcal{T}^*(z(t))| = 0. \tag{3.31}$$

In practice, the estimate $\hat{x}(t) = \mathcal{T}^*(z(t))$ of $x(t)$ is obtained by solving (3.28) numerically from an arbitrary initial condition $z(0)$, which is not necessarily $\mathcal{T}(x(0))$ as $x(0)$ is generally unknown in a state estimation context. Therefore, there is an initial transient period during which $z(t)$ is not yet close to $\mathcal{T}(x(t))$ hence the estimated state $\hat{x}(t)$ is not close to $x(t)$. The length of this transient depends on $\lambda_{\min}$, exponentially for z, but in a nonlinear fashion that depends on $\mathcal{T}^*$ for $\hat{x}$.

Remark 1. *According to this result, $z \in \mathbb{C}^{d_y(d_x+1)}$. Therefore, in order to represent this filter with real numbers only, we need $d_z = 2d_y(d_x+1)$. However, in practice, we assume that the $d_y(d_x+1)$ complex eigenvalues needed for D are complex conjugates, such that we only need dimension $d_z = d_y(d_x+1)$ to represent the real filter $z \in \mathbb{R}^{d_z}$.*

This problem formulation in which one seeks a transformation to a normal system form, for which existence and regularity results are known, is common in systems theory. Therefore, results on this topic may enable extensions to other, similar contexts. For example, many works on learning nonlinear SSMs rely on a coordinate transformation from the observed coordinates to some suitable latent coordinates, which is often approximated by an autoencoder [29], [93].

Remark 2. *For any ODE of general form $\dot{x}(t) = f(t, x(t))$ where f is continuously differentiable ($\mathcal{C}^1$), there is a unique maximal solution [113]. It is then possible to run the ODE "in backward time" or equivalently "backward in time", i.e., write out the solution from a given instant t_0 in decreasing time, with the change of variables $y(t') = x(-t)$, i.e., $\dot{y}(t') = -f(t', y(t'))$. This can become useful for KKL observers, e.g., for estimating the initial condition $x(0)$ instead of the current state $x(t)$ by running the observer backward in time for long enough.*

3.4.2 Extensions to nonautonomous systems

These results are extended to nonautonomous systems in [114]. The system equations are then

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x, u) \end{aligned} \tag{3.32}$$

where $u \in \mathcal{U}$ is the input of dimension d_u. Assumption 2 naturally extends to nonautonomous systems if it is true for any fixed input u of interest. The following theorem proves the existence of a KKL observer in the nonautonomous case.

Theorem 2 (Bernard and Andrieu [114])**.** *Take some fixed input $u \in \mathcal{U}$. Suppose Assumptions 1 and 2 hold for this u with a certain $\bar{t}_u \geq 0$. Define $d_z = d_y(d_x+1)$. Then, there exists a set S of zero measure in $\mathbb{C}^{d_z}$ such that for any diagonalizable matrix $D \in \mathbb{C}^{d_z \times d_z}$ with eigenvalues $(\lambda_1, \ldots, \lambda_{d_z})$ in $\mathbb{C}^{d_z} \setminus S$ with $\Re\lambda_i < 0$, $i \in \{1, \ldots, d_z\}$, and any $F \in \mathbb{C}^{d_z \times d_x}$ such that (D, F) is controllable, there exists a mapping $\mathcal{T}_u : \mathbb{R} \times \mathbb{R}^{d_x} \to \mathbb{C}^{d_z}$ that satisfies the following equation on $\mathcal{X}$*

$$\frac{\partial \mathcal{T}_u}{\partial x}(t, x) f(x, u(t)) + \frac{\partial \mathcal{T}_u}{\partial t}(t, x) = D\mathcal{T}_u(t, x) + Fh(x, u(t)), \tag{3.33}$$

and a mapping $\mathcal{T}_u^* : \mathbb{R} \times \mathbb{C}^{d_z} \to \mathbb{R}^{d_x}$ such that $\mathcal{T}_u(t, \cdot)$ and $\mathcal{T}_u^*(t, \cdot)$ only depend on the past values of u on $[0, t]$, and $\mathcal{T}_u(t, \cdot)$ is injective $\forall\, t \geq \bar{t}_u$ with a left-inverse $\mathcal{T}_u^*(t, \cdot)$ on $\mathcal{X}$. Then, the trajectories of (3.26) remaining in $\mathcal{X}$ and any trajectory of

$$\dot{z} = Dz + Fy, \tag{3.34}$$

satisfy

$$|z(t) - \mathcal{T}_u(t, x(t))| \leq M|z(0) - \mathcal{T}_u(0, x(0))|e^{-\lambda_{min} t} \tag{3.35}$$

for some $M > 0$ and with

$$\lambda_{min} = \min\left\{|\Re\lambda_1|, \ldots, |\Re\lambda_{d_z}|\right\}. \tag{3.36}$$

This yields

$$\lim_{t\to+\infty} |x(t) - \mathcal{T}_u^*(t, z(t))| = 0. \tag{3.37}$$

The main differences with the autonomous case are that the eigenvalues no longer need to be sufficiently large and that $\mathcal{T}$ is time-dependent. Crucially, it depends on the entire history of the input $u([0,t])$ and becomes injective for $t \geq \bar{t}_u$ with $\bar{t}_u$ from the backward distinguishability assumption. This rapidly leads to numerical intractability if no analytical expression of $\mathcal{T}_u$ is available.

A potential solution is to consider inputs structured by an ODE and rely on functional observers [115]. Assume an autonomous system is only backward distinguishable with respect to a function of the state $q(x)$, i.e., $q(x_a(t)) \neq q(x_b(t))$ implies that there exists $s \in [t-\bar{t}, t]$ such that $h(x_a(s)) \neq h(x_b(s))$ as in Assumption 2. Then, a KKL observer of that function can be built as in Sect. 3.4.1, i.e., it can be shown that there exists a continuous function $\mathcal{T}$ that transforms x into z and a globally defined, uniformly continuous map τ such that $\tau(\mathcal{T}(x)) = q(x)$. We then have

$$\lim_{t\to+\infty} |q(x(t)) - \tau(z(t))| = 0. \tag{3.38}$$

It is shown in [115] that the previous results for autonomous KKL observers extend to this case, i.e., with static mappings $\mathcal{T}$ and τ.

Consider again the nonautonomous system (3.32). Assume the input u can be generated by an auxiliary dynamical system of state $\omega \in \mathbb{R}^{d_\omega}$, so that we have the complete system

$$\begin{aligned} \dot{\omega} &= l(\omega), \qquad u = s(\omega) \\ \dot{x} &= f(x,u), \qquad y = h(x,u). \end{aligned} \tag{3.39}$$

This leads to the functional observer described in [115, Sect. III], where $X = (x, \omega)$ is the extended state, $Y = (y, u)$ is the extended output, and $q(X) = x$ is the functional w.r.t. which the system is distinguishable. Hence, an autonomous KKL observer can be designed for (3.39) as in Sect. 3.4.1, and we have

$$\lim_{t\to+\infty} |x(t) - \tau(z(t))| = 0 \tag{3.40}$$

where

$$\dot{z} = Dz + FY. \tag{3.41}$$

When a finite-dimensional of the input is known, a static KKL observer can thus be designed, at the cost of a higher dimension $d_z = (d_y + d_u)(d_x + d_\omega + 1)$.

3.4.3 Numerical KKL observers

The previous results show the existence of the transformations $\mathcal{T}$ and $\mathcal{T}^*$, proving the global convergence of the corresponding observer under weak observability assumptions. However, in general, it is not possible to compute these transformations analytically. Hence, it has been proposed to approximate them numerically with

neural networks [116]. We refer to these as numerical KKL observers; the aim is to *learn to observe*, i.e., assuming the dynamics are known, learn to estimate the state from measurements. This is done in a supervised fashion, by generating simulations of the full state of the plant/observer system.

Starting from a grid of points in $\mathcal{X}$, the system is simulated forward in time for "long enough", i.e., until it can be considered that the observer has converged. The trajectory in z is simulated jointly for a given pair (D, F), so that a dataset of points (x_i, z_i), $i \in \{1, \dots, N\}$ is obtained. The mappings $\mathcal{T} : x \mapsto z$ and $\mathcal{T}^* : z \mapsto x$ are approximated with two feed-forward NNs based on this dataset. The learned observer can then be applied to estimate the full state x from experimental measurements y.

In the autonomous case, this simply consists in simulating z forward in time for a new output trajectory and computing the corresponding estimated state $\hat{x} = \mathcal{T}^*(z)$. As for the nonautonomous case, the authors of [116] focus on input-affine systems of form

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{3.42}$$

and apply a methodology proposed in [114]. In this special case, one can generate the training set with a nominal control input $u^0(\cdot)$, learn the transformations, then simulate the corresponding observer with any other input $u(\cdot)$ as

$$\dot{z} = Dz + Fy + \frac{\partial \mathcal{T}}{\partial x}(t, \mathcal{T}^*(t, z))g(\mathcal{T}^*(t, z))(u - u^0). \tag{3.43}$$

If the observer converges sufficiently fast compared to the Lipschitz constant of the extra term in $\dot{z}$, then it is shown in [114], [116] that the previous results extend and the corresponding estimate $\hat{x} = \mathcal{T}^*(t, z)$ converges asymptotically to the true state.

From the seminal paper [116], this line of work has since been extended, i.e., to discrete-time systems [117] or by improving the performance [118]. We further build on it in Ch. 7.

Part I

Nonlinear observer theory for dynamics model learning

This thesis is motivated by the growing topic of digital twins, which has recently gained interest at Ansys. A digital twin is a numerical replica of a physical object, able to simulate its behavior with high fidelity. This enables investigating the dynamic evolution of the system at a lower cost, in order to ease the design, examine different scenarios, or evaluate the influence of certain parameters over time, for example how wear affects the system.

For digital twins to reach and maintain the desired level of accuracy over time, they need to be regularly updated with experimental data from the physical system. However, simulation models are often complex and high-dimensional, while the experimental data is often partial and noisy, as not all aspects of the dynamics can be measured accurately. Therefore, refining the dynamics model with experimental data is both a relevant and challenging task. In this first part of the thesis, we leverage concepts from observer design to enable system identification with machine learning. We first focus on systems in the observable canonical form in Ch. 4, and interconnect a high-gain observer with a Gaussian process model of the dynamics, for which joint convergence of the state and dynamics estimation can be shown. We then consider a more general setting in Ch. 5, and combine neural ordinary differential equations with recognition models based on Kazantis-Kravaris / Luenberger observers to learn a flexible state-space model from partial observations.

Chapter 4

Joint state and dynamics estimation in the observable canonical form

Résumé Dans ce chapitre, une forme particulière de système est considérée : la forme canonique observable, et un algorithme d'estimation de l'état et de la dynamique de façon conjointe est formulé. Pour cela, nous proposons d'interconnecter un observateur grand gain et un processus gaussien approximant la dynamique. L'observateur fournit des estimations d'état, qui servent de données d'apprentissage pour le modèle dynamique. Le modèle, mis à jour à son tour, est ensuite utilisé pour améliorer l'observateur. La convergence conjointe de l'observateur et du modèle dynamique est prouvée pour un gain suffisamment élevé, aux perturbations près. L'apprentissage simultané de la dynamique et l'estimation de l'état sont illustrés dans des simulations d'un oscillateur non linéaire et d'un système masse-ressort-masse. Une comparaison quantitative avec les travaux antérieurs, des améliorations de la méthode et une extension pour aborder un cas d'usage impliquant un modèle de simulation Ansys sont ensuite présentées.

Abstract In this chapter, we focus on a particular type of system: the observable canonical form, and jointly perform state and dynamics estimation. We propose interconnecting a high-gain observer and a dynamics learning framework, specifically a Gaussian process model. The observer provides state estimates, which serve as the training data for the dynamics model. The updated model, in turn, is used to improve the observer. Joint convergence of the observer and the dynamics model is proved for high enough gain, up to the measurement and process perturbations. Simultaneous dynamics learning and state estimation are demonstrated in simulations of a nonlinear oscillator and a mass-spring-mass system. A quantitative comparison to previous work, improvements of the method, and an extension to tackle a use case involving an Ansys simulation model are then presented.

Parts of this chapter are published in the IEEE Control Systems Letters under the title *Joint State and Dynamics Estimation with High-Gain Observers and Gaussian Process Models* [49].

4.1 Introduction

In this chapter, we propose an approach combining nonlinear state estimation and Gaussian processes to learn the dynamics of a system in observable canonical form from partial observations. On the one hand, reconstructing the full state of a system from noisy, partial measurements falls into the area of state estimation and observer design. On the other hand, most existing approaches for learning dynamics models require knowing the full state [14]. Therefore, joint state estimation and dynamics learning are needed, a problem often referred to in the machine learning community as inference and learning [70].

The design of observers for nonlinear systems is a complex task for which various approaches have been investigated (see [42], [43] for an overview). We focus on the fairly large class of systems that can be expressed in the so-called observable canonical form (see Sec. 4.2). For these systems, one can design high-gain observers (HGOs), which rely on a triangular structure with increasing gain power to compensate for the nonlinearity farther from the measurement. HGOs have been used for a wide variety of applications [119], [120]. In particular, they provide robustness to model uncertainty, as practical convergence can be proved for high enough gain, given only an upper bound on the nonlinearity [44].

Learning dynamics models is also an active research topic. In particular, Gaussian process (GP) state-space models are increasingly used [24], [68]. These nonparametric models exhibit many advantageous properties for learning dynamical systems: they are flexible, data-efficient, probabilistic, and can easily incorporate prior knowledge (see Sec. 3.1 and [53] for details). Thanks to their analytical formulation, GPs also allow for theoretical guarantees, see e.g., [67], [78], which is often desirable for control applications.

The problem of joint inference and learning for GP state-space models is tackled in its most general form in [70] using the expectation-maximization algorithm. In the first step, measurements are collected and the posterior distribution of the GP is computed. In the second step, all hyperparameters, including the pseudo inputs and outputs representing the evolution of the latent states, are optimized to maximize the data log-likelihood. Improvements of this approach have been proposed, e.g., by shifting to variational inference [80] while incorporating additional structure [81], [82], [84]; see Sec. 3.1 for more details. However, the optimization procedure remains high-dimensional and nonconvex. This leads to a high computational burden and a risk of overfitting, which can make the models difficult to train. Furthermore, no theoretical guarantees are yet provided for such methods.

Recent works tackle this problem by combining observer design and data-driven dynamics learning with universal approximators. The model is learned with a neural network using smooth, continuous-time weight update laws [41], [121] or a basis expansion [122], then incorporated into an observer built as a copy of the system with added linear output injection terms. Limited theoretical guarantees have been shown [41], [122], but joint convergence has only been proved if suitable gains can be found by solving a large set of linear matrix inequalities [121]. However, this yields an unusual neural network model for f and an observer with a high number of parameters left to tune, limiting the practical use of the framework.

In this chapter, we combine the predictive power of machine learning with existing convergence results for state estimation. Our main contribution is the design of a framework for simultaneous state and dynamics estimation, by combining an HGO that estimates the full state from measurements and a GP model that learns the unknown nonlinearity. Convergence guarantees for both the observer and the dynamics model are provided; practical applicability is discussed and demonstrated in simulations. This builds upon the scheme proposed in [123], in which the nonlinearity is considered as a state with partially known dynamics in an extended HGO, and learned by an identifier satisfying certain requirements. The key difference of our approach is to directly learn a discrete model of the nonlinearity instead of differentiating it and extending the observer. This enables considering controlled systems and input-dependent nonlinearities, and decreasing the error in the data used for regression. Furthermore, it reduces the dimensionality of the observer, which attenuates noise amplification by the HGO. We also show that more flexible, non-parametric

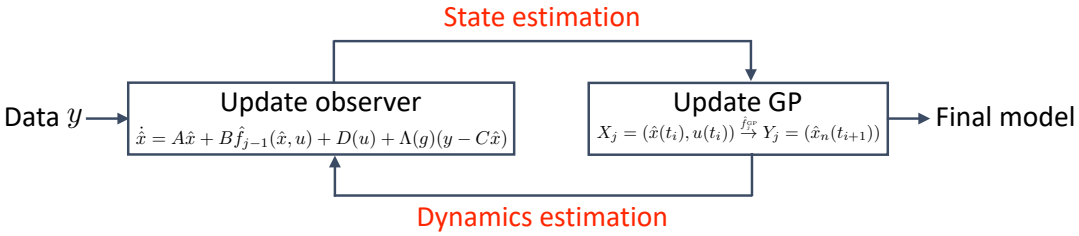


Figure 4.1: Structure of the framework: after each cycle, an updated model $\hat{f}_j$ is computed, and the observer is adapted.

models such as GPs can learn the input-dependent dynamics while satisfying the smoothness assumptions which are necessary to prove joint convergence.

After formalizing the problem, we present the proposed framework in Sec. 4.2. In Sec. 4.3 we show joint convergence of both state and dynamics estimation, then demonstrate our approach on numerical examples in Sec. 4.4. A comparison to related work is given in Sec. 4.5 and improvements to the method in Sec. 4.6. We propose an extension in Sec. 4.7, motivated by an Ansys use case. Finally, we discuss the limitations in Sec. 4.8, before concluding in Sec. 4.9.

4.2 Problem formulation and proposed framework

We consider a dynamical system with state $x \in \mathbb{R}^{d_x}$, output $y \in \mathbb{R}$, and control input $u \in \mathbb{R}^{d_u}$ bounded at each time step, where $d_x, d_u \in \mathbb{N}$. For ease of notation, we focus on the single-output case, but all results extend to multiple outputs by concatenation. We assume the following observable canonical form

$$\begin{aligned} \dot{x} &= Ax + Bf(x, u) + D(u) + d, \\ y &= Cx + \epsilon \end{aligned} \tag{4.1}$$

with f an unknown nonlinearity acting on the last state x_d, while the rest of the dynamics follows a chain of integrators:

$$A = \begin{pmatrix} \mathbb{O}_{d_x-1} & \mathbb{I}_{d_x-1} \\ 0 & \mathbb{O}_{d_x-1}^\top \end{pmatrix}, \qquad B = \begin{pmatrix} \mathbb{O}_{d_x-1} \\ 1 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 \\ \mathbb{O}_{d_x-1} \end{pmatrix}^\top.$$

The input function $D : \mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$ is continuous and known while $d \in \mathbb{R}^{d_x}$, $\epsilon \in \mathbb{R}$ are unknown disturbances, typically considered deterministic (see Remark 4). All vectors are column vectors; $\mathbb{O}_{d_x}$ denotes a vector of d_x zeros, while $\mathbb{I}_{d_x}$ is the identity matrix of size d_x. A broad class of systems can be transformed into this canonical form without knowing f, e.g., all differentially observable systems [42, Sec. 7.1]. We aim to compute an estimate $\hat{x}$ of the full state from measurements y, while jointly learning a model $\hat{f}$ of f. We make the following assumptions on (4.1).

Assumption 3. *The true nonlinearity f is Lipschitz continuous of constant L_f. There exist compact sets $\mathcal{X}$ and $\mathcal{U}$ such that $x(t) \in \mathcal{X}$, $u(t) \in \mathcal{U}$ $\forall\, t \geq 0$.*

Since f is continuous on a compact space, its absolute value on $\mathcal{X} \times \mathcal{U}$ is also bounded by $f_{\max}$.

The proposed observer follows a cyclic structure, illustrated in Figure 4.1. During cycle number $j \in \mathbb{N}^*$, the observer produces an estimated state trajectory based on

measurements and on the current dynamics model $\hat{f}_{j-1}$. This data is sampled and saved. At the end of the cycle, the model is updated based on the available estimated data. It produces an estimate $\hat{f}_j$, which is then used by the observer for the next cycle.

4.2.1 High-gain observer

During cycle number j, the HGO performs state estimation using the current model of the dynamics $\hat{f}_{j-1}$:

$$\dot{\hat{x}} = A\hat{x} + B\hat{f}_{j-1}(\hat{x}, u) + D(u) + \Lambda(g)(y - C\hat{x}). \tag{4.2}$$

The gain is denoted $g > 1$, while $\Lambda(g)$ is the gain matrix following a standard high-gain construction:

$$\Lambda(g) := \begin{pmatrix} gL_1 & g^2L_1 & \cdots & g^{d_x}L_{d_x} \end{pmatrix}, \tag{4.3}$$

with $L = \begin{pmatrix} L_1 & \cdots & L_{d_x} \end{pmatrix} \in \mathbb{R}^{d_x}$ such that $A - LC$ is Hurwitz. Equation (4.2) corresponds to the observer block in Figure 4.1. We make the following assumption, which is ensured by our dynamics model described hereafter.

Assumption 4. *For all $j \in \mathbb{N}$, $\hat{f}_j$ is continuous and its norm is bounded by $\hat{f}_{\max}$.*

Hence, we can pick $\mathcal{X}$ large enough such that $\hat{x}(t) \in \mathcal{X}\ \forall\, t \geq 0$. With Assumptions 3–4, the error on the nonlinearity $\hat{f}(x, u) - f(x, u)$ is bounded. Then, as proved in [44] and used in the literature on HGOs, practical convergence in finite time can be shown in the absence of disturbances: for a given error level $\nu > 0$ and a given time $\bar{t} > 0$, there exists a gain g high enough to ensure that for all $t \geq \bar{t}$, $|\hat{x}(t) - x(t)| \leq \nu$. Hence, no matter how bad the approximation $\hat{f}$ of f is, as long as an upper bound of the difference is known, the practical convergence of the observer can be guaranteed for high enough gain. We leverage this property to build our method.

4.2.2 Reminder on Gaussian processes

In this chapter, we focus on Gaussian processes (GPs). However, any learning algorithm that satisfies our assumptions detailed in Sec. 4.3 can be used.

Recall the preliminaries on GPs presented in Sec. 3.1. We simply remind the reader that when f is a GP, given a dataset (X, Y) with Gaussian noise of variance $\sigma_\epsilon^2 > 0$ on the output, the prediction $f(x)$ at an unobserved point x is normally distributed with posterior mean and variance

$$\mu(x|X, Y) = \underline{k}(x)^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1} Y \tag{4.4}$$

$$\sigma^2(x|X, Y) = k(x, x) - \underline{k}(x)^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1} \underline{k}(x), \tag{4.5}$$

where $\mathbb{I}$ is the identity matrix of suitable dimension, $K = (k(x_i, x_j))_{x_i, x_j \in X}$ is the covariance matrix of X, and $\underline{k}(x) = (k(x_i, x))_{x_i \in X}$. The kernel k usually depends on some hyperparameters, which are considered fixed in this work.

Assumption 5. *The kernel k is positive definite (see Sec. 3.3), Lipschitz continuous of constant L_k, and its norm is bounded by $k_{\max}$:*

$$\left|k(x, y) - k(x', y)\right| \leq L_k \left|x - x'\right| \tag{4.6}$$

and $|k(x, y)| \leq k_{\max}$ for any (x, y) in $\mathcal{X}^2$.

This is the case for most commonly used covariance functions, such as the squared exponential. GPs are increasingly used for learning dynamics thanks to their flexibility, data efficiency, and analytical formulation. In this work, we follow the deterministic interpretation (see Sec. 3.1) and use the GP posterior mean as a function approximator to estimate f.

4.2.3 Learning method

The state trajectory estimated by the observer is used to learn the dynamics model $\hat{f}$ through batch updates. For each update indexed by $j \in \mathbb{N}^*$, a dataset of length N is constructed by sampling this trajectory with period Δt, starting from the last sample collected at t_j:

$$\begin{aligned} X_j &= \begin{pmatrix} \hat{x}(t_{j,N})^\top & u(t_{j,N})^\top \\ \cdots & \\ \hat{x}(t_{j,1})^\top & u(t_{j,1})^\top \end{pmatrix} \in \mathbb{R}^{N \times (d_x + d_u)} \\ Y_j &= \begin{pmatrix} \hat{x}_d(t_{j,N-1}) & \cdots & \hat{x}_d(t_j) \end{pmatrix} \in \mathbb{R}^N \end{aligned} \tag{4.7}$$

where x_d denotes the last dimension of x, and we denote $t_j - N\Delta t$ the time N samples ago by $t_{j,N}$. Since the nonlinearity acts on the last dimension, x_d is the only output that needs to be collected. The size N of the dataset acts as a moving window of length N over the estimated trajectory. The j^{th} update $\hat{f}_j$ is learned from inputs X_j and outputs Y_j, then used in the observer (4.2) for the next cycle. It can be updated offline since this may necessitate more computing power, and the updates are not necessarily periodic. The model learned from (4.7) is

$$\mu_j(\cdot | X_j, Y_j) = \underline{k}_j(\cdot)^\top (K_j + \sigma_\epsilon^2 \mathbb{I}_N)^{-1} Y_j \tag{4.8}$$

with $K_j = (k(x_i, x_l))_{x_i, x_l \in X_j}$, $\underline{k}_j(x) = (k(x_i, x))_{x_i \in X_j}$. For the formulation of the GP posterior and the theoretical guarantees, Gaussian measurement noise of variance $\sigma_\epsilon^2 > 0$ is assumed. However, this need not be the case in reality as we only use the GP as a function approximator and σ_ϵ^2, similarly to other hyperparameters, is chosen in practice for calibration purposes.

We perform nonlinear regression to estimate the mapping $\mu : (\hat{x}, u) \mapsto \hat{x}_d(t + \Delta t)$. However, $\hat{f}$ in the observer corresponds to the continuous time derivative of $\hat{x}_d$. Hence, we form $\hat{f}_j$ with a Euler differentiation step:

$$\hat{f}_j(\hat{x}, u) = \frac{1}{\Delta t}(\mu_j(\hat{x}, u | X_j, Y_j) - \hat{x}_d). \tag{4.9}$$

There are numerically more advantageous schemes for numerical differentiation, e.g., central differences. However, these require querying the GP model at different time steps, e.g., $(\hat{x}(t - \Delta t/2), u(t - \Delta t/2))$. Therefore, we do not implement them in this chapter, but expect that enabling such schemes would improve the performance of the method.

To guarantee the boundedness of $\hat{f}$, we saturate it directly in the observer by imposing $\left|\hat{f}\right| \leq \hat{f}_{\max}$. The model (4.9) corresponds to the GP block in Figure 4.1; given the continuity of (4.8) and this saturation, it satisfies Assumption 4.

Remark 3. *The choice of Δt results from a trade-off: small enough to keep the numerical error from (4.9) low, large enough to see a real difference between two samples. Learning a discrete*

model from the estimated trajectory prevents us from building an extended state observer, contrarily to [123].

4.3 Theoretical guarantees

As stated previously, HGOs are robust to model uncertainty for systems in the observable canonical form. This enables us to decouple the procedures of state estimation and dynamics learning, as detailed above. Indeed, thanks to this robustness, convergence guarantees for the observer can still be obtained even in the worst-case scenario, i.e., with maximal but bounded model error, at the cost of a high gain. These convergence properties are then transferred to the dynamics model through its smoothness w.r.t. the dataset used for learning. Both practical and asymptotic convergence results are provided for the complete estimation scheme.

In the following proofs, we focus on the ℓ_2 norm for vectors and matrices, denoted by $|\cdot|$, but equivalent bounds can be obtained for any vector norm and its induced matrix norm.

4.3.1 Smoothness of GP models

We first give a technical result on the smoothness of GP models, showing that the posterior mean is Lipschitz continuous not onlyw.r.t. the test point but also w.r.t. the training dataset.

Lemma 1. *Under Assumptions 3–5, the dynamics model $\hat{f}$ as defined in (4.9) is Lipschitz continuous with respect to each of its variables: $(x,u) \mapsto \hat{f}(x,u|X,Y)$ with constant L_x, and $(X,Y) \mapsto \hat{f}(x,u|X,Y)$ with constant L_z.*

Proof. The Lipschitz continuity of the GP posterior mean and variance w.r.t. a test point are given in [67, Th. 3.1]; we derive this result again for the mean in our notations. We fix two test points $\underline{x}, \underline{x}' \in \mathcal{X} \times \mathcal{U}$, where $\underline{x} = (x,u)$ is the extended state, the dataset (X,Y) used for learning, and a kernel k which is L_k-Lipschitz. When the dataset (X,Y) is fixed, we omit the dependency on it for simplicity. We have:

$$\left|\mu(\underline{x}) - \mu(\underline{x}')\right| \leq \left|\underline{k}(\underline{x}) - \underline{k}(\underline{x}')\right| \left|(K + \sigma_\epsilon^2 \mathbb{I})^{-1} Y\right| \tag{4.10}$$

where μ is the mean of the GP learned on X, Y with k. The covariance matrix K is symmetric, positive semidefinite by definition, hence $\tilde{K} = K + \sigma_\epsilon^2 \mathbb{I}$ is symmetric, positive definite given $\sigma_\epsilon^2 > 0$. Therefore, $\tilde{K}$ is invertible and $\tilde{K}^{-1}$ is diagonalizable with eigenvalues upper bounded by $1/\sigma_\epsilon^2$, which yields

$$\left|\tilde{K}^{-1}\right| \leq \frac{1}{\sigma_\epsilon^2}. \tag{4.11}$$

Since all N terms of Y live on a compact space, there exists $Y_{\max} > 0$ such that $|Y_i| \leq Y_{\max}\ \forall\ i \in 1, ..., N$, hence $|Y| \leq Y_{\max}\sqrt{N}$. Since the covariance function is L_k-Lipschitz, we have $|\underline{k}(\underline{x}) - \underline{k}(\underline{x}')| \leq L_k\sqrt{N}|\underline{x} - \underline{x}'|$, and we obtain

$$\left|\mu(\underline{x}) - \mu(\underline{x}')\right| \leq \frac{L_k}{\sigma_\epsilon^2} Y_{\max} N \left|\underline{x} - \underline{x}'\right| = L_x' \left|\underline{x} - \underline{x}'\right| \tag{4.12}$$

as in [67]. Adding the Euler differentiation step proves the first claim of the lemma:

$$\begin{aligned}\left|\hat{f}(\underline{x}) - \hat{f}(\underline{x}')\right| &= \left|\frac{1}{\Delta t}(\mu(\underline{x}) - x_d) - \frac{1}{\Delta t}(\mu(\underline{x}') - x'_d)\right| \\ &\leq \frac{L'_x + 1}{\Delta t}\left|\underline{x} - \underline{x}'\right| = L_x\left|\underline{x} - \underline{x}'\right|, \end{aligned} \tag{4.13}$$

where x_d is the last entry of x in $\underline{x} = (x, u)$.

The Lipschitz continuity of the GP posterior w.r.t. the dataset is less often considered, though it is investigated in [69, Th. 1] for the posterior variance.

Consider two datasets (X, Y) and (X', Y') such that at least one entry differs between X and X' resp. Y and Y', the corresponding vectors $\underline{k}(\cdot)$ and $\underline{k}'(\cdot)$ and covariance matrices K and K' as defined in Sec. 4.2.2, and a fixed test point $\underline{x}$. We have

$$\left|\mu(\underline{x}|X, Y) - \mu(\underline{x}|X', Y')\right| \leq \left|\mu(\underline{x}|X, Y) - \mu(\underline{x}|X, Y')\right| + \left|\mu(\underline{x}|X, Y') - \mu(\underline{x}|X', Y')\right|. \tag{4.14}$$

The first term can be explicitly written as

$$\begin{aligned}\left|\mu(\underline{x}|X, Y) - \mu(\underline{x}|X, Y')\right| &= \left|\underline{k}(\underline{x})^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1}(Y - Y')\right| \\ &\leq \frac{k_{\max}}{\sigma_\epsilon^2}\sqrt{N}\left|Y - Y'\right| \end{aligned} \tag{4.15}$$

with the Cauchy-Schwartz inequality, the bound on the absolute value of $k(\cdot, \cdot)$ and (4.11). The second term can again be separated as

$$\begin{aligned}\left|\mu(\underline{x}|X, Y') - \mu(\underline{x}|X', Y')\right| \leq &\left|\underline{k}(\underline{x})^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1} Y' - \underline{k}'(\underline{x})^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1} Y'\right| \\ &+ \left|\underline{k}'(\underline{x})^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1} Y' - \underline{k}'(\underline{x})^\top (K' + \sigma_\epsilon^2 \mathbb{I})^{-1} Y'\right|. \end{aligned} \tag{4.16}$$

Due to the previous bounds and the Lipschitz continuity of the kernel (Assumption 5), with $\tilde{K}' = K' + \sigma_\epsilon^2 \mathbb{I}$, this first term can be bounded as

$$\begin{aligned}\left|\underline{k}(\underline{x})^\top \tilde{K}^{-1} Y' - \underline{k}'(\underline{x})^\top \tilde{K}^{-1} Y'\right| &\leq \frac{Y_{\max}}{\sigma_\epsilon^2}\sqrt{N}\left|\underline{k}(\underline{x}) - \underline{k}'(\underline{x})\right| \\ &\leq \frac{Y_{\max}}{\sigma_\epsilon^2}\sqrt{N} L_k \left|X - X'\right|_F \\ &\leq \frac{Y_{\max}}{\sigma_\epsilon^2}\sqrt{N(d_x + d_u)} L_k \left|X - X'\right|, \end{aligned} \tag{4.17}$$

where $|\cdot|_F$ is the Frobenius norm and $(d_x + d_u)$ is the number of columns of X. The second term in (4.16) can be bounded as

$$\left|\underline{k}'(\underline{x})^\top (K + \sigma_\epsilon^2 \mathbb{I})^{-1} Y' - \underline{k}'(\underline{x})^\top (K' + \sigma_\epsilon^2 \mathbb{I})^{-1} Y'\right| \leq k_{\max} Y_{\max} N \left|\tilde{K}^{-1} - \tilde{K}'^{-1}\right|. \tag{4.18}$$

We then use a modified form of Hua's identity on matrices [124]. Assuming all necessary matrices are invertible, the standard form of this identity is

$$A - (A^{-1} + (B^{-1} - A)^{-1})^{-1} = ABA. \tag{4.19}$$

Rearranging the terms and taking the inverse yields

$$(B^{-1} - A)^{-1} = -A^{-1} + (A - ABA)^{-1}. \tag{4.20}$$

Then, making the change of variables from B^{-1} to B and multiplying by -1 gives

$$(A - B)^{-1} = A^{-1} - (A - AB^{-1}A)^{-1}, \tag{4.21}$$

and factorizing the last term by $B - A$ on the left side gives

$$(A - B)^{-1} = A^{-1} + A^{-1}B(A - B)^{-1}. \tag{4.22}$$

Set $A = K + \sigma_\epsilon^2 \mathbb{I} = \tilde{K}$, and $B = \tilde{K} - \tilde{K}'$ a perturbation of A. Since $\tilde{K}$ and $\tilde{K}'$ are symmetric, positive definite, A and $A - B$ are invertible. Using (4.22) with this choice of A, B yields:

$$\tilde{K}^{-1} - \tilde{K}'^{-1} = -\tilde{K}^{-1}(K - K')\tilde{K}'^{-1}. \tag{4.23}$$

Then, we have with basic norm inequalities and using the Lipschitz continuity of k, by denoting $\bar{K} = (k(x_i, x_j) - k(x_i', x_j))_{i,j\in\{1,\dots,N\}}$:

$$\begin{aligned}
\left|K - K'\right| &\leq \left|K - \bar{K}\right|_F + \left|\bar{K} - K'\right|_F \\
&\leq \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}(k(x_i, x_j) - k(x_i', x_j))^2} + \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}(k(x_i', x_j) - k(x_i', x_j'))^2} \\
&\leq 2L_k\sqrt{N\sum_{i=1}^{N}\left|x_i - x_i'\right|^2} \leq 2L_k\sqrt{N}\left|X - X'\right|_F \\
&\leq 2L_k\sqrt{(d_x + d_u)N}\left|X - X'\right|.
\end{aligned} \tag{4.24}$$

Hence, we obtain by replacing (4.23) into (4.18) and using the previous bound

$$\left|\underline{k}'(\underline{x})^\top \tilde{K}^{-1}Y' - \underline{k}'(\underline{x})^\top \tilde{K}'^{-1}Y'\right| \leq \frac{2}{\sigma_\epsilon^4}k_{\max}Y_{\max}L_k N^{3/2}\sqrt{(d_x + d_u)}\left|X - X'\right|. \tag{4.25}$$

Putting (4.15), (4.17) and (4.25) together yields

$$\begin{aligned}
\left|\mu(\underline{x}|X, Y) - \mu(\underline{x}|X, Y')\right| &\leq \frac{k_{\max}}{\sigma_\epsilon^2}\sqrt{N}\left|Y - Y'\right| + \frac{Y_{\max}}{\sigma_\epsilon^2}L_k\sqrt{N(d_x + d_u)}\left|X - X'\right| \\
&\quad + \frac{2}{\sigma_\epsilon^4}k_{\max}Y_{\max}L_k N^{3/2}\sqrt{(d_x + d_u)}\left|X - X'\right| \\
&:= L_z'(\left|X - X'\right| + \left|Y - Y'\right|).
\end{aligned} \tag{4.26}$$

Adding the Euler differentiation step concludes the proof:

$$\begin{aligned}
\left|\hat{f}(\underline{x}|X, Y) - \hat{f}(\underline{x}|X', Y')\right| &= \left|\frac{1}{\Delta t}(\mu(\underline{x}|X, Y) - x_d) - \frac{1}{\Delta t}(\mu(\underline{x}|X', Y') - x_d)\right| \\
&\leq \frac{L_z'}{\Delta t}(\left|X - X'\right| + \left|Y - Y'\right|) \\
&:= L_z(\left|X - X'\right| + \left|Y - Y'\right|),
\end{aligned} \tag{4.27}$$

where x_d is the last entry of x in $\underline{x} = (x, u)$. □

Since the dataset (X_j, Y_j) is constructed from state estimation samples, the error in this data directly depends on the state estimation error. This corresponds to the stability requirement in [123]. Then, Lemma 1 guarantees that any state estimation error in the dataset is smoothly transferred to the obtained model. This is essential to obtain stability guarantees and corresponds to the regularity requirement in [123]. While we focus on GPs, any learning algorithm satisfying Lemma 1 based on a dataset constructed as in (4.7) can be used in our framework, to produce an observer and a dynamics model that can both be used for further control tasks.

4.3.2 Practical convergence

We denote:

$$\begin{aligned} \bar{d} &= \begin{pmatrix} g^{-1}d_1 & \cdots & g^{-d_x}d_{d_x} \end{pmatrix} \\ \underline{x} &= (x, u) \\ \underline{\hat{x}} &= (\hat{x}, u) \\ |x|_{\bar{t}} &= \max\{|x(t)|, t \leq \bar{t}\} \\ \limsup_{t+j\to\infty} h_j(x(t)) &= \lim_{t\to\infty} \lim_{j\to\infty} \left(\sup_{l\geq j} h_l(x(t))\right) \end{aligned} \tag{4.28}$$

for a sequence of functions $\{h_l\}$, $l \in \mathbb{N}$. At a given $j \in \mathbb{N}^*$, $\hat{f}_j^*$ is the nonlinearity that would have been learned if the true data X_j^*, Y_j^* had been available, i.e., if the state $x(t)$ had been directly available for sampling according to (4.7) instead of its estimate $\hat{x}(t)$. The prediction error is written as

$$\varepsilon_j(\cdot) = \hat{f}_j(\cdot) - f(\cdot), \tag{4.29}$$

while $\varepsilon_j^*(\cdot) = \hat{f}_j^*(\cdot) - f(\cdot)$ is the optimal prediction error that would have been obtained if the true state had been available instead of an estimate. We now state a first result on the practical convergence of the proposed framework.

Theorem 3. *For system (4.1)–(4.2) with (4.9) under Assumptions 3–5, for any given error level $\nu > 0$, any time $\bar{t} > 0$, there exists a gain g^* large enough such that for all $g \geq g^*$, $t \geq \bar{t}$ and $j \in \mathbb{N}^*$ such that $\bar{t} \leq t_j \leq t$, in the absence of disturbances ($d \equiv \epsilon \equiv 0$), we have for any fixed $\underline{x} \in \mathcal{X} \times \mathcal{U}$:*

$$\max\left\{|\hat{x}(t) - x(t)|, \left|\hat{f}_j(\underline{x}) - \hat{f}_j^*(\underline{x})\right|\right\} \leq \nu. \tag{4.30}$$

Proof. The proof follows three steps.

Step 1. During a given cycle indexed by some fixed $j \in \mathbb{N}^*$, with $t_j \geq t \geq t_{j-1}$, the model error at any point $\underline{x}$ can be bounded by the maximal error:

$$\left|\hat{f}_{j-1}(\underline{x}) - f(\underline{x})\right| \leq \hat{f}_{\max} + f_{\max}. \tag{4.31}$$

Hence, the scaled error dynamics denoted by $\zeta(t, j) = \mathrm{col}(g^{1-i}(\hat{x}_i(t) - x_i(t)))$, $i \in \{1, ..., d_x\}$ follow a triangular structure suitable for the use of high-gain observers:

$$\dot{\zeta}(t, j) = g(A - LC)\zeta(t, j) + g^{1-d_x}B\left(\hat{f}_{j-1}(\hat{x}, u) - f(x, u)\right) - H(g)d + gL\epsilon, \tag{4.32}$$

where $H(g) = \begin{pmatrix} 1 & \cdots & g^{1-d_x} \end{pmatrix}$ and $A - LC$ is Hurwitz. As shown in standard practical convergence proofs for HGOs [44], [123], this scaled error system obtained for the HGO combined with Assumptions 3 – 4 enables the proof of practical convergence. For all $j \in \mathbb{N}^*$, no matter the current estimation $\hat{f}_{j-1}$, there exists $g^* > 1$ and ρ_0, ρ_1, $\rho_1 > 0$ such that $\forall\, g \geq g^*$:

$$\begin{aligned} |\hat{x}_i(t) - x_i(t)| \leq & \max\left\{g^{i-1}\rho_0 e^{-\rho_1 g t}|\hat{x}(0) - x(0)|, \rho_1 g^{i-d_x-1}, \rho_2 g^{i-1}\left|(\bar{d}, \epsilon)\right|_t\right\} \\ & := M_i(t), \end{aligned} \tag{4.33}$$

where $\bar{d} = \text{col}(g^{-i}d_i)$, $i \in \{1, ..., d_x\}$. Notice that $M_i(t) < M_j(t)$ if $i < j$ for $g > 1$ and t fixed, and that for fixed i, the first term in $M_i(t)$ decreases as t increases, while the last term increases.

Step 2. The key then lies in bounding the error on X_j and Y_j as

$$\begin{aligned} \left|X_j - X_j^*\right| \leq \left|X_j - X_j^*\right|_F &= \sqrt{\sum_{l=1}^{N}\sum_{i=1}^{d_x}\left|\hat{x}_i(t_{j,l}) - x_i(t_{j,l})\right|^2} \\ &\leq \sqrt{\sum_{l=1}^{N}\sum_{i=1}^{d_x} M_i(t_{j,l})^2} \leq \sqrt{d_x N}\bar{M}_d(t_{j,N}, t_{j,1}), \end{aligned} \tag{4.34}$$

$$\begin{aligned} \left|Y_j - Y_j^*\right| \leq \left|Y_j - Y_j^*\right|_F &= \sqrt{\sum_{l=0}^{N-1}\left|\hat{x}_d(t_{j,l}) - x_d(t_{j,l})\right|^2} \\ &\leq \sqrt{\sum_{l=0}^{N-1} M_{d_x}(t_{j,l})^2} \leq \sqrt{N}\bar{M}_d(t_{j,N-1}, t_j), \end{aligned} \tag{4.35}$$

where $\bar{M}_d(t, t') = \max\left\{g^{d_x-1}\rho_0 e^{-\rho_1 g t}|\hat{x}(0) - x(0)|, \frac{\rho_1}{g}, \rho_2 g^{d_x-1}\left|(\bar{d}, \epsilon)\right|_{t'}\right\}$. This boils down to showing that the error on the input and output datasets used to learn $\hat{f}_j$ decreases as the state estimation error decreases, with a delay of N sampling time steps corresponding to the time before earlier samples with a larger error are forgotten.

Step 3. Notice that the first term in $\bar{M}_d(t, t')$ decreases when t increases, while the third increases when t' increases. With the Lipschitz-continuity assumption on $\hat{f}_j$, this yields

$$\left|\hat{f}_j(\underline{x}) - \hat{f}_j^*(\underline{x})\right| \leq L_z\left(\left|X_j - X_j^*\right| + \left|Y_j - Y_j^*\right|\right) \leq L_z\sqrt{N}(1 + \sqrt{d_x})\bar{M}_d(t_{j,N}, t_j). \tag{4.36}$$

Combining (4.33) and (4.36) as such leads to a joint practical convergence result, with an additional term corresponding to the disturbances. In the absence of disturbances, i.e., with $d \equiv \epsilon \equiv 0$, this concludes the proof.

□

4.3.3 Asymptotic convergence

Theorem 3 shows that the practical convergence guarantees obtained for HGOs with bounded nonlinearity extend to the complete error system. Both the state estimation error and the error made by the dynamics model due to seeing only estimated instead

of true data can be made arbitrarily small arbitrarily fast, up to the disturbances, at the cost of a high gain. We now present an asymptotic convergence result arising from the practical convergence of HGOs in the presence of model uncertainty, by bounding this uncertainty depending on the data, the test point, and the optimal ε^*.

Theorem 4. *For system (4.1)–(4.2) with (4.9) under Assumptions 3–5, there exist constants $c, c' > 0$ and a gain $g^* > 0$ large enough such that $\forall\, g \geq g^*$, $\forall\, i \in \{1, ..., d_x\}$ and for any fixed test point $\underline{x} \in \mathcal{X} \times \mathcal{U}$, we have*

$$\limsup_{t\to\infty} |\hat{x}_i(t) - x_i(t)| \leq \max\left\{ g^{i-d_x} \limsup_{t+j\to\infty} \left|\varepsilon_j^*(x,u)\right|, cg^{i-1} \limsup_{t\to\infty} \left|(\bar{d},\epsilon)\right| \right\}, \tag{4.37}$$

$$\limsup_{t+j\to\infty} \left|\hat{f}_j(\underline{x}) - \hat{f}_j^*(\underline{x})\right| \leq c' \max\left\{ \limsup_{t+j\to\infty} \left|\varepsilon_j^*(x,u)\right|, g^{d_x-1} \limsup_{t\to\infty} \left|(\bar{d},\epsilon)\right| \right\}. \tag{4.38}$$

Proof. The proof follows the same line of thought as the proof of Theorem 3. First, using definition (4.29) and Lemma 1, we show that for any fixed $j \in \mathbb{N}^*$, $(x, \hat{x}, u) \in \mathcal{X} \times \mathcal{X} \times \mathcal{U}$, there exists $c_1 > 0$ such that

$$\begin{aligned} \left|\hat{f}_{j-1}(\hat{x},u) - f(x,u)\right| &\leq \left|\hat{f}_{j-1}(\hat{x},u) - \hat{f}_{j-1}^*(x,u)\right| + \left|\varepsilon_{j-1}^*(x,u)\right| \\ &\leq c_1\Big(|\hat{x} - x| + \left|X_{j-1} - X_{j-1}^*\right| + \left|Y_{j-1} - Y_{j-1}^*\right| + \left|\varepsilon_{j-1}^*(x,u)\right| \Big). \end{aligned} \tag{4.39}$$

Step 1. During a given cycle indexed by some fixed $j \in \mathbb{N}^*$, $t_j \geq t \geq t_{j-1}$, with x, $\hat{x}$, u the values of the state, estimated state and control input of system (4.1)–(4.2) at t, we have the same error dynamics

$$\dot{\zeta}(t,j) = g(A - LC)\zeta(t,j) + g^{1-d_x} B\big(\hat{f}_{j-1}(\hat{x},u) - f(x,u)\big) + H(g)d + gL\epsilon \tag{4.40}$$

with $\zeta(t,j) = \mathrm{col}(g^{1-i}(\hat{x}_i(t) - x_i(t)))$ and $H(g) = \begin{pmatrix} 1 & \cdots & g^{1-d_x} \end{pmatrix}$.

By definition of ζ and assuming $g > 1$, we have $|\hat{x} - x| \leq g^{d_x-1}|\zeta|$. With Lyapunov arguments and replacing (4.39) into (4.40), the linear part of the system is stable: taking $V(\zeta) = \frac{1}{2}|\zeta|^2$, we have $\dot{V}(\zeta) \leq (g\lambda_{\max}(A - LC) + c_1)V(\zeta)$ where the maximum eigenvalue of a matrix M is denoted by $\lambda_{\max}(M)$, and since $A - LC$ is Hurwitz, stability is obtained for g large enough. Thus, taking the other terms into account similarly to [123] and with standard high-gain arguments as presented in [44], there exists $c_i > 0$, $i \in \{2, ..., 6\}$, $g_1 > 1$ high enough such that $\forall\, g \geq g_1$

$$\begin{aligned} |\zeta(t,j)| \leq \max\Big\{ & c_2 e^{-c_3 g(t-t_{j-1})} \left|\zeta(t_{j-1},j)\right|, c_4 g^{-d_x} \left|\varepsilon_{j-1}^*(x,u)\right|, c_5 \left|(\bar{d},\epsilon)\right|_t, \\ & c_6 g^{-d_x}\left(\left|X_{j-1} - X_{j-1}^*\right| + \left|Y_{j-1} - Y_{j-1}^*\right| \right) \Big\}. \end{aligned} \tag{4.41}$$

Step 2. Going to the limit in time and in number of cycles and using (4.34)-(4.35), we see that there exists $c_7 > 0$ such that

$$\limsup_{t+j\to\infty} \left(\left|X_j - X_j^*\right| + \left|Y_j - Y_j^*\right| \right) \leq c_7 g^{d_x-1} \limsup_{t+j\to\infty} |\zeta(t,j)|. \tag{4.42}$$

Therefore, using the definition of lim sup, we have by injecting (4.42) into (4.41) that there exists $g_2 > \max\{c_4, c_6c_7\}$ such that $\forall\, g \geq g_2$:

$$\limsup_{t+j\to\infty}|\zeta(t,j)| \leq \max\left\{g^{1-d_x}\limsup_{t+j\to\infty}\left|\varepsilon_j^*(x,u)\right|, c_5\limsup_{t\to\infty}\left|(\bar{d},\epsilon)\right|\right\}. \tag{4.43}$$

This directly yields the first claim

$$\limsup_{t\to\infty}|\hat{x}_i(t) - x_i(t)| \leq \max\left\{g^{i-d_x}\limsup_{t+j\to\infty}\left|\varepsilon_j^*(x,u)\right|, c_5 g^{i-1}\limsup_{t\to\infty}\left|(\bar{d},\epsilon)\right|\right\}. \tag{4.44}$$

Step 3. Lemma 1 yields the second claim for any fixed test point $\underline{x} \in \mathcal{X}\times\mathcal{U}$

$$\limsup_{t+j\to\infty}\left|\hat{f}_j(\underline{x}) - \hat{f}_j^*(\underline{x})\right| \leq c_8 \max\left\{\limsup_{j\to\infty}\left|\varepsilon_j^*(x,u)\right|, g^{d_x-1}\limsup_{t\to\infty}\left|(\bar{d},\epsilon)\right|\right\}, \tag{4.45}$$

with $c_8 > 0$. Using the same error decomposition as (4.39) also yields:

$$\begin{aligned}\limsup_{t+j\to\infty}\left|\hat{f}_j(\underline{x}) - f(\underline{x})\right| \leq & c_8c_9 \max\left\{\limsup_{j\to\infty}\left|\varepsilon_j^*(x,u)\right|, g^{d_x-1}\limsup_{t\to\infty}\left|(\bar{d},\epsilon)\right|\right\} \\ & + c_9\limsup_{j\to\infty}\left|\varepsilon_j^*(\underline{x})\right|\end{aligned} \tag{4.46}$$

with $c_9 > 0$.

□

Theorem 4 bounds the difference between $\hat{f}$ and $\hat{f}^*$ that would have been obtained using true instead of estimated data for learning, without any assumption on the fit of the GP. If ε^* is zero, both the state estimation and the prediction error are input-to-state stable w.r.t. the disturbances. If there are also no disturbances, they converge asymptotically. Since GPs with universal kernels are universal function approximators, ε^* can get very small (it converges up to the numerical errors due to the Euler differentiation) as the number of samples grows to infinity, if the samples are densely distributed over $\mathcal{X}\times\mathcal{U}$, i.e., if the state-action space is well explored. However, if the hyperparameters of the GP do not enable a good fit or if the data is not rich enough, then ε^* may be large. We further note that it is also possible to bound $\left|\hat{f}_j(\underline{x}) - f(\underline{x})\right|$ similarly to (4.38) using the same error decomposition as (4.39): a term $\varepsilon_j^*(\underline{x})$ appears, representing the optimal error due to the GP model at the test point $\underline{x}$. This is derived in (4.46).

Remark 4. *Nowhere do we use the form of d and ϵ. In principle, they could also be realizations of stochastic processes. However, Theorem 4 may then not be as meaningful as, depending on the process,* lim sup *may not exist.*

Remark 5. *The proposed framework is modular: any regressor $\hat{f}$ can be used seamlessly instead of the GP. If Assumptions 3–4 and Lemma 1 are satisfied, the same theoretical guarantees will hold. This is the case for models of the form $\hat{f}(\cdot) = \theta^\top\sigma(\cdot)$, where θ is a parameter to be learned from data following a Lipschitz procedure (such as recursive least squares), and σ is a known, Lipschitz continuous feature map. Other observer designs*

providing guarantees similar to HGOs could also be considered, e.g., sliding mode observers [125]. Conversely, some machine learning techniques such as neural networks do not exhibit similar regularity properties as they result from a nonconvex optimization procedure.

Remark 6. *In practice, one can learn the residual model* $\mu_{res} : (\hat{x}, u) \mapsto \hat{x}_d(t + \Delta t) - \hat{x}_d - \Delta t D_{d_x}(u)$ *instead of* μ*, and use* $\mu(\hat{x}, u) = \mu_{res}(\hat{x}, u) + \hat{x}_d + \Delta t D_{d_x}(u)$ *for prediction. This eases the training process by incorporating prior knowledge into the regression problem, while minimally changing the form of* Y_j*, which leads to a factor* $(\sqrt{2} + \sqrt{d_x})$ *instead of* $(1 + \sqrt{d_x})$ *in* (4.36) *and maintains the theoretical guarantees.*

4.4 Numerical simulations

We now demonstrate the proposed approach with numerical simulations of two nonlinear systems. For each experiment, we go over ten cycles, i.e., we run the HGO for a certain time given measurements, then update the GP model with the estimated trajectory, include the new GP into the HGO, and run again ten times.

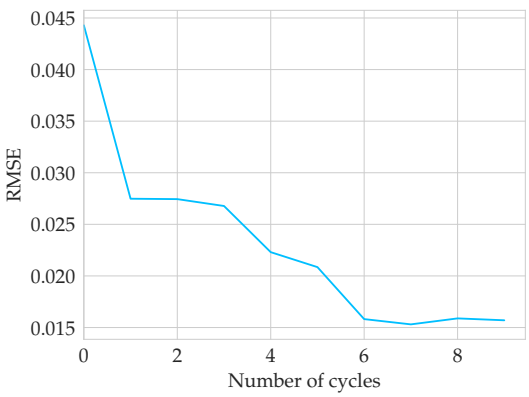
4.4.1 Duffing oscillator

We start by evaluating the proposed framework on simulations of the Duffing oscillator. The dynamics of this two-dimensional system can be described as follows, with $x = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^\top$:

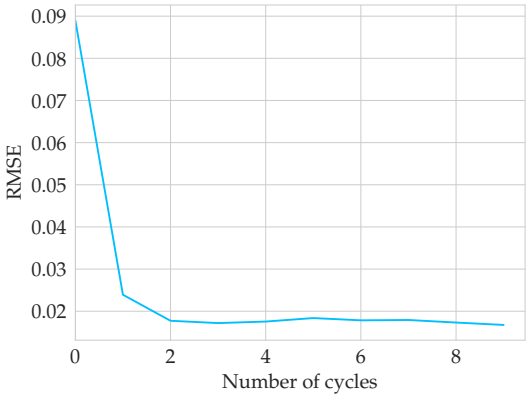
$$\begin{aligned} \dot{x} &= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x - \begin{pmatrix} 0 \\ 1 \end{pmatrix} (\alpha x_1 + \beta x_1^3 + \delta x_2) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} x + \epsilon. \end{aligned} \tag{4.47}$$

The system is already in the observable canonical form. We simulate it with a Runge-Kutta method of order 5(4) (RK4/5) with adaptive step size [1], for 10 cycles of 30 seconds gathering 500 samples per cycle ($\Delta t = 0.06s$). The observer receives measurements corrupted by Gaussian measurement noise of variance $\sigma_\epsilon^2 = 10^{-5}$ (no process noise) and estimates the full state trajectory, which is then used by the GP for learning. The dataset is built according to (4.7). We set $\alpha = -1$, $\beta = 1$, $\delta = 0.3$, $u(t) = 0.4\cos(1.2t)$, $N = 3000$, $g = 8$, $L = (5, 5)^\top$. We use a squared exponential kernel with fixed hyperparameters, whose values have been optimized over a subset of data at the beginning of the experiment, leading to a scaling parameter 110 and lengthscales $(5, 15, 150)$.

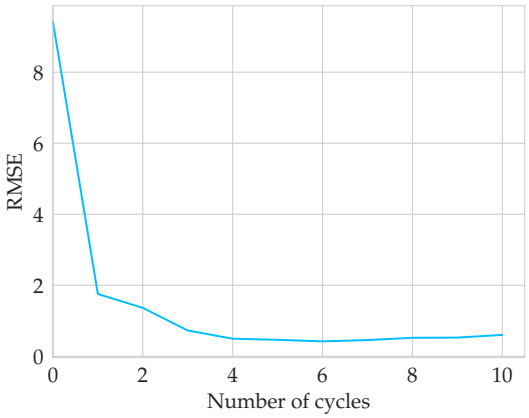
At the end of each cycle, we evaluate the current model on ten test trajectories of length 18 s with random initialization: three obtained using random control, four using $u(t) = 0.4\cos(1.2t)$, and three with $u(t) = 0$. These test trajectories are either used for *open loop predictions* (from the correct initial condition, the current dynamics model predicts the trajectory) or *closed loop estimations* (the current dynamics model is used inside the HGO to estimate the trajectory given the measurement and an arbitrary initial condition). We refer to these evaluations as open loop resp. closed loop test trajectories, or as predictions resp. estimations. We also compute the error of the learned model μ compared to the ground truth over a random grid of points, along with the estimation error of the observer over the last cycle. For these different metrics, we compute the root mean squared error (RMSE) over a given sequence of



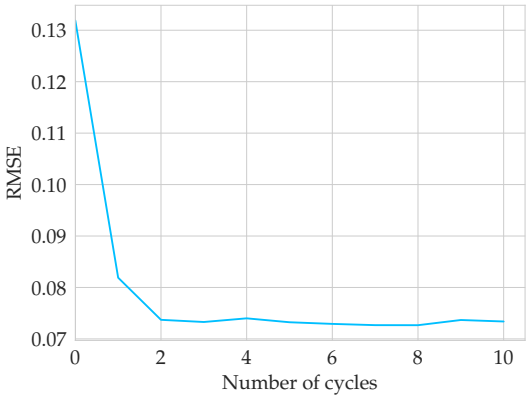
(a) GP predictions over a grid



(b) State estimation over the last cycle



(c) Ten random open loop test trajectories



(d) Ten random closed loop test trajectories

Figure 4.2: Learning the dynamics of the Duffing oscillator (4.47): different error metrics over ten cycles.

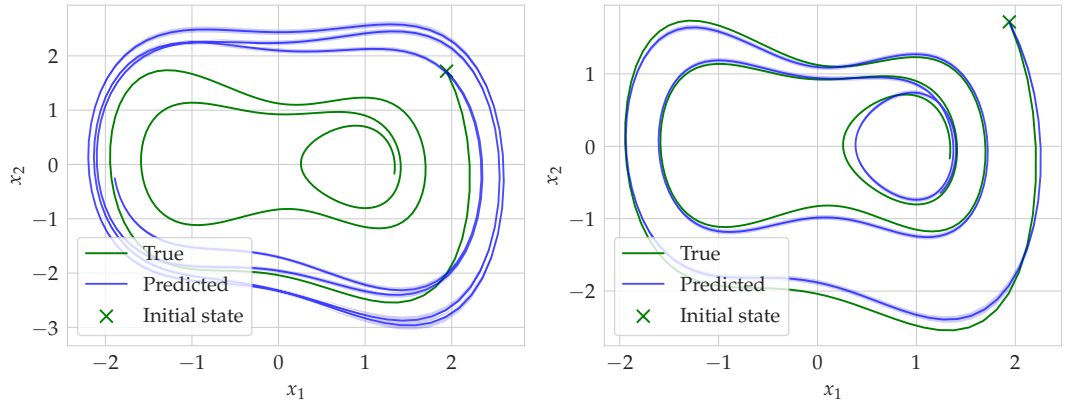
time instants t_i with $i \in \{1, ..., N\}$, as

$$RMSE(\hat{x}, x) := \sqrt{\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{d_x} (x_j(t_i) - \hat{x}_j(t_i))^2} \tag{4.48}$$

where x is the ground truth and $\hat{x}$ is the estimated value to be compared.

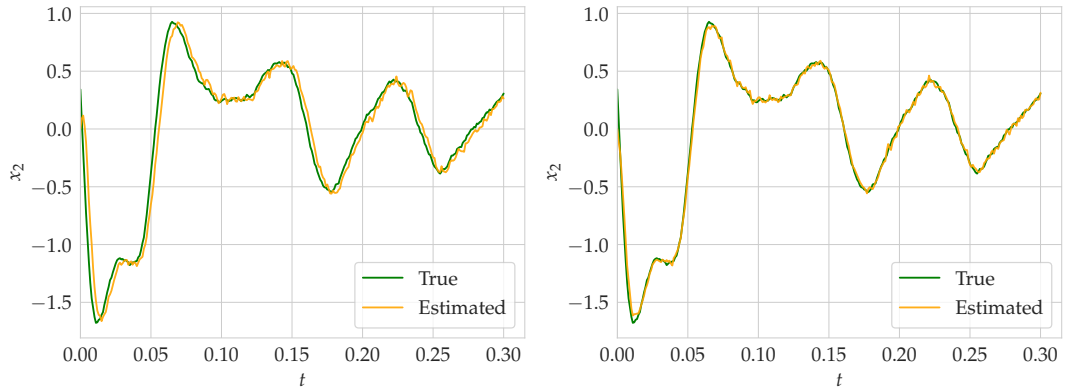
Examples of the obtained results are provided in Figures 4.2–4.4. We observe that all considered metrics decrease over the cycles of state and dynamics estimation. In particular, the dynamics model is accurate enough after ten cycles to predict a test trajectory despite the chaotic nature of the system, as seen in Fig. 4.3: the prediction error accumulates over time as expected, but stays low enough to attain accurate predictions. The improved dynamics model also leads to improved state estimation, as seen in Fig. 4.4, where the delay in the prediction has been corrected.

Remark 7. *We follow the deterministic view (see Sec. 3.1.2) and focus on the GP posterior mean, used as a function approximator for f (up to the numerical differentiation step). In the following plots, we only show the variance at each deterministic point in the trajectory for illustration purposes, all computations focus on the mean.*



(a) After one cycle (b) After ten cycles

Figure 4.3: Phase portrait of the Duffing oscillator (4.47), for an open loop test trajectory after one and ten cycles.



(a) Before learning (b) After ten cycles

Figure 4.4: True and estimated trajectory of x_2 for the Duffing oscillator (4.47), for a closed loop test trajectory before any learning and after ten cycles.

Remark 8. *In our code, we simulate all continuous-time dynamics such as HGOs with an RK4/5 solver. However, to simulate the open loop test trajectories, we mix an explicit Euler step with a direct query of the discrete GP model, as follows:*

$$\begin{aligned} x(t+\Delta t) &= \begin{pmatrix} 1 & \dots & 1 & 0 \end{pmatrix} * \Big(x(t) + \Delta t \big(Ax(t) + D(u(t)) \big) \Big) + B\mu(x(t), u(t)) \\ &= x(t) + \Delta t \big(Ax(t) + D(u(t)) \big) + B\mu_{res}(x(t), u(t)) \end{aligned} \tag{4.49}$$

where μ is the current GP model, μ_{res} the current residual model, and $*$ denotes the element-wise product. This enables directly using the GP model instead of differentiating it numerically to simulate continuous-time dynamics. For fairness, all open loop test trajectories with all methods are predicted using explicit Euler. This is enough for our academic example, however, we would expect higher performance with a more advanced numerical solver such as RK4/5 for all continuous parts, combined with a discrete model when needed.

Figure 4.5: Diagram of the mass-spring-mass system.

4.4.2 Nonlinear mass-spring-mass system

We also demonstrate the performance of the proposed approach on a mass-spring-mass system with a nonlinear spring, as illustrated in Figure 4.5, and motivated by series-elastic actuators, e.g., [126]. We assume the system can be described by

$$\begin{aligned} m_1 \ddot{x}_1 &= f_k(x_2 - x_1) \\ m_2 \ddot{x}_2 &= -f_k(x_2 - x_1) + u, \end{aligned} \tag{4.50}$$

where x_1, x_2 are the positions of the two objects, $m_1 = m_2 = 1$ are their masses, and $f_k(\cdot)$ is some unknown nonlinear function representing the spring dynamics.

Assuming system (4.50) is differentially observable of order 4 (see [42] and references therein), it can be transformed into the observable canonical form (4.1). This is done by introducing a new state $z \in \mathbb{R}^4$, taking $z_1 = x_1$ and computing the successive derivatives of z_1. As in (4.1), this yields

$$\begin{aligned} \dot{z} &= Az + Bf(z,u) + d \\ y &= Cz + \epsilon, \end{aligned} \tag{4.51}$$

where f is an unknown nonlinearity. The proposed approach can directly be applied to (4.51) without further knowledge about f. In our simulations, we use $f_k(\cdot) = k_1(\cdot) + k_2(\cdot)^3$ with $k_1 = 0.3$, $k_2 = 0.1$ the spring constants. This yields for the true system in observable canonical form:

$$f(z,u) = \frac{3k_2}{m_1 m_2}(u - (m_1+m_2)z_3)v_1^2 + \frac{6k_2}{m_1}v_1 v_2^2 + \frac{k_1}{m_1 m_2}(u - (m_1+m_2)z_3), \tag{4.52}$$

$$\alpha = \sqrt[3]{\frac{m_1 z_3}{2k_2} + \sqrt{\left(\frac{k_1}{3k_2}\right)^3 + \left(\frac{m_1 z_3}{2k_2}\right)^2}},$$

$$\beta = \sqrt[3]{\frac{m_1 z_3}{2k_2} - \sqrt{\left(\frac{k_1}{3k_2}\right)^3 + \left(\frac{m_1 z_3}{2k_2}\right)^2}},$$

$$v_1 = \alpha + \beta, \qquad v_2 = \frac{z_4}{\frac{k_1}{m_1} + \frac{3k_2}{m_1}v_1^2}.$$

We simulate (4.51) with d, ϵ Gaussian noise of standard deviation $\sigma_d = \sigma_\epsilon = 10^{-4}$ for ten cycles of 15 seconds each, sampled at $\Delta t = 0.06s$. We set $u(t) = 0.4\cos(1.2t)$, $N = 3000$, $g = 10$, $L = (5,5,3,1)^\top$, and $\hat{f}_0 \equiv 0$. We use a squared exponential kernel whose hyperparameters are fixed by maximizing the marginal log-likelihood on a subset of data offline, leading to a scaling parameter 3.5 and lengthscales $(150, 150, 1.5, 2.5, 2.5)$. We run 10 experiments from 10 random initial conditions with $x_1 \in [0, 0.1]$, $x_2 \in [0.1, 0.2]$, and $\dot{x}_1, \dot{x}_2 \in [-0.005, 0.005]$[1]. For each, we start by precomputing a grid of random states and inputs, along with 50 test trajectories of

[1]Code to reproduce the results is available at github.com/monabf/obsGP_recogNODE.git.

12 s, using a random initial state and one of three control strategies: random control, $u(t) = 0.4\cos(1.2t)$, or $u(t) = 0$.

As for the Duffing oscillator, we evaluate both the observer and the model at the end of every cycle, for each experiment. The observer (4.2) containing the current model is evaluated by computing the RMSE between true and estimated trajectory over the last cycle, but also over the test trajectories, given $\hat{z}(t_0) = (y(t_0), 0, 0, 0)^\top$ and $y(t)$. The model (4.9) is evaluated by computing the RMSE of one step ahead predictions over the precomputed grid, and the RMSE of the predicted test trajectories, given the initial state but no measurements.

We observe joint convergence of the observer and the dynamics model for all experiments. The error in all considered metrics decreases over time, as depicted in Figure 4.6. The numbers themselves are not necessarily meaningful, but their decreasing behavior and the visual results before and after a few cycles are significant. The variance is due to having different initial conditions, evaluation grids, and test trajectories for each run rather than a different performance of the method. The remaining error is caused by the measurement and process noise, along with the irreducible model error given the available data. A test trajectory as estimated by the observer is presented in Figure 4.7. Before the model is learned Fig.4.7a, the observer's estimates are delayed compared to the true state, because the observer waits for correction from the measurements. Once the model has been learned Fig.4.7b, the observer can anticipate and produce accurate estimates without delay. The phase portrait of another test trajectory predicted by the dynamics model is also depicted in Figure 4.8. It shows the final model can predict the first 100 time steps accurately, then deviates.

4.5 Comparison to previous work

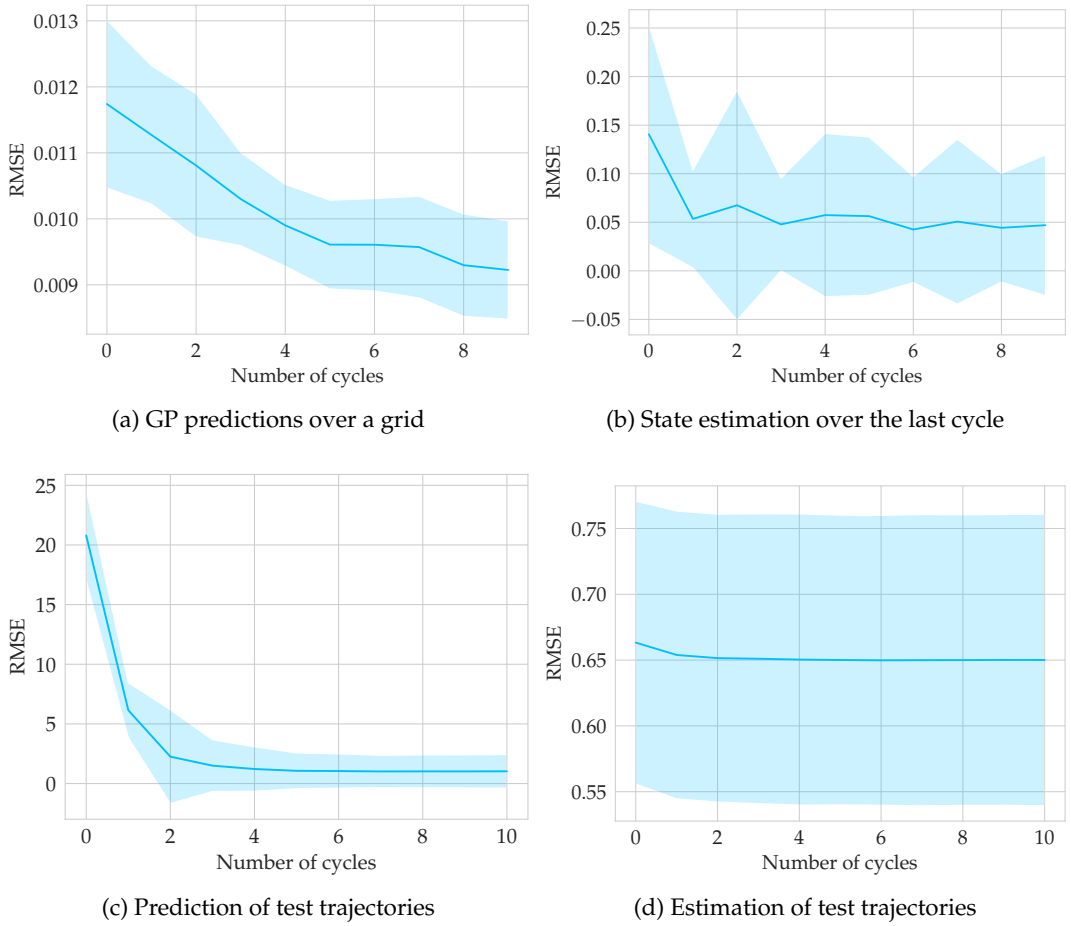
The proposed framework is inspired by [123]. In this section, we recall the main idea of this alternative approach, qualitatively describe its main characteristics, then illustrate them with numerical simulations. In the following, we denote the method presented in Sec. 4.2 as method 1, which learns a model $\hat{f}$. The framework presented in [123] is referred to as method 2, which learns a model $\hat{\phi}$.

4.5.1 Summary of method 2

In [123], the analysis is restricted to autonomous systems. We slightly extend it to allow for comparison with method 1, and consider a system in observable canonical form with an unknown nonlinearity, as in (4.1):

$$\dot{x} = Ax + Bf(x) + D(u), \tag{4.53}$$

where it is assumed that f does not depend on u. The unknown nonlinearity is represented by a parametric model $\hat{\phi}(x)$, whose parameters θ need to be estimated. This model can be a least squares identifier, e.g., $\hat{\phi}(x) = \theta^\top \sigma(x)$ where $\sigma(\cdot)$ is a known feature vector, a wavelet decomposition, or any other type of parametric model. The objective is the same as in this chapter: jointly estimating the state and the nonlinearity. However, the implementation varies. In [123], the nonlinear part of the dynamics $f(x)$ is approximated by adding an extra state ξ, which is estimated together with the underlying state x by an extended observer, i.e., an HGO with one



(a) GP predictions over a grid

(b) State estimation over the last cycle

(c) Prediction of test trajectories

(d) Estimation of test trajectories

Figure 4.6: RMSE of metrics over 10 simulations of the mass-spring-mass system (4.51) (mean $\pm$ 2 standard deviations).

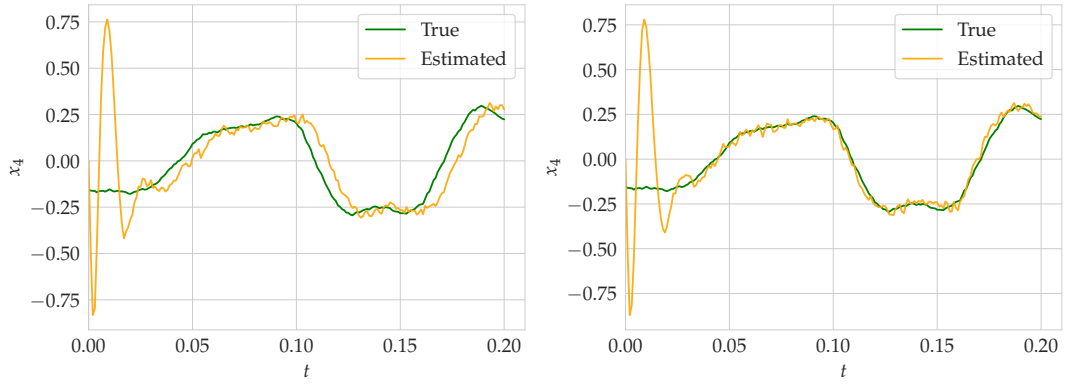
more state compared to (4.2). The dynamics of this observer are given by

$$\begin{aligned}\dot{\hat{x}} &= A\hat{x} + B\xi + D(u) + \Lambda_1(g)(y - C\hat{x}) \\ \dot{\xi} &= \frac{\partial \hat{\phi}}{\partial x}(\hat{x})(A\hat{x} + B\xi + D(u)) + \Lambda_2(g)(y - C\hat{x})\end{aligned} \tag{4.54}$$

where $\hat{\phi}$ is the current model of the nonlinearity and

$$\Lambda(g) := \begin{pmatrix} gL_1 & g^2L_1 & \cdots & g^{d_x}L_{d_x} & g^{d_x+1}L_{d_x+1} \end{pmatrix}^\top = \begin{pmatrix} \Lambda_1(g) \\ \Lambda_2(g) \end{pmatrix}. \tag{4.55}$$

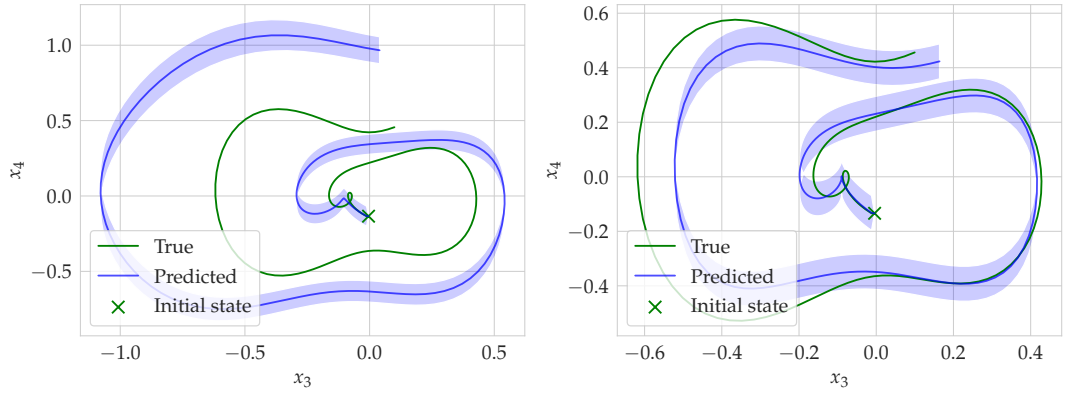
Similarly to method 1, the robustness of HGOs against model errors is leveraged to show that both $\hat{x}$ and ξ converge practically to their true value for g high enough. As in method 1, the estimated state trajectories are used cyclically to update the model of the nonlinearity $\hat{\phi}$. However, method 2 learns a model $\hat{\phi} : \hat{x} \mapsto \xi$ by adapting the parameters θ, with an adaptation law that is smooth w.r.t. the estimated trajectories (regularity requirement, corresponding to our Lemma 1). For example, a recursive least squares procedure falls under this category: θ is selected to minimize $\sum_i \left|\theta^\top \sigma(\hat{x}(t_i)) - \xi(t_i)\right|$ over the available samples i. Similarly to Th. 4, it is then shown that for g high enough, $\hat{x}$ converges asymptotically to the true state while θ converges



(a) Before learning

(b) After ten cycles

Figure 4.7: Estimation of a test trajectory of z_4 (random control).



(a) After one cycle

(b) After ten cycles

Figure 4.8: Prediction of a test trajectory of z_4 against z_3 with sine control (mean $\pm$ 2 standard deviations).

to its optimal value given the form of the model $\hat{\phi}$.

4.5.2 Trade-offs

Method 1 presented in Sec. 4.2 and method 2 presented in [123] are similar, but exhibit different trade-offs. In this section, we make these explicit before illustrating them with numerical results.

The main difference between methods 1 and 2 lies in how the data is collected for learning the dynamics model. In method 2, the nonlinearity is considered as an extra state and the HGO is extended to estimate it in continuous time, then a continuous dynamics model $\hat{\phi} : \hat{x} \mapsto \hat{\xi}$ is learned. This regression problem is formulated directly in continuous time. However, there are several limitations:

- A state ξ has been added to the HGO, so that it has one extra dimension compared to method 1. Hence, measurement noise is even more amplified on ξ which is then used for learning $\hat{\phi}$, so that the obtained model is more sensitive to noise.

- The extended HGO (4.54) includes the derivative of $\hat{\phi}$ in its dynamics. However, the derivative of an approximation is often a poor fit for the true derivative, so this term possibly contains a lot of error.
- Because $\hat{\phi}$ is differentiated to be used in (4.54), it cannot include a dependency in u. In fact, in [123], the framework is discussed only for autonomous systems. We have added the term $D(u)$, which does not influence the rest of the method, but $\hat{\phi}$ being independent of u remains a strong restriction, most of all for systems that need to be transformed into the observable canonical form so that f in the transformed coordinates will depend on u in many cases.

On the contrary, in method 1, the HGO only estimates $\hat{x}$, and learns a discrete model $\mu : \hat{x} \mapsto \hat{x}_d(t + \Delta t)$. This more direct procedure, without extending the observer, avoids the previous issues by learning a discrete model directly from estimates of x. Hence, it usually leads to a more accurate dynamics model and therefore more accurate open loop predictions. However, if this discrete model needs to be used in a continuous-time scheme, for example for simulating it inside a numerical solver with a different time step, or using it inside a continuous-time observer, then it has to be numerically differentiated as in (4.9). For example, at the end of each cycle, we update the continuous-time HGO (4.2) with a Euler differentiation step:

$$\dot{\hat{x}} = A\hat{x} + B\frac{\mu_j(\hat{x}, u) - \hat{x}_d}{\Delta t} + D(u) + \Lambda(g)(y - C\hat{x}) \tag{4.56}$$

where μ_j is the current GP model. This adds a numerical error that deteriorates the performance of the continuous model compared to its discrete counterpart. However, it would be possible to use higher order numerical schemes such as central differences by retaining the predictions of the GP at different time steps, which would reduce the numerical error. We leave this open for future work.

4.5.3 Numerical illustration

We now illustrate the previous points with numerical simulations. Note that comparing both methods quantitatively is not straightforward. Hence, the following numerical results are mere illustrations of the trade-offs faced by both approaches rather than thorough quantitative comparisons. In particular, the observer has an extra dimension for method 2, so that the gain matrices

$$A - L_1 C \tag{4.57}$$

$$\begin{pmatrix} A & B \\ \mathbb{O}_n^\top & 0 \end{pmatrix} - L_2 \begin{pmatrix} C & 0 \end{pmatrix} \tag{4.58}$$

for methods 1 and 2 respectively do not have the same dimension. There is no simple method to choose these gains so that both designs have "the same" tuning and can be compared objectively. Therefore, we tune them by hand such that the resulting HGO has similar performance for both methods. This leads to $L_1 = (5, 5)$ resp. $L_2 = (5, 5, 1)$ with $g = 8$ for the Duffing oscillator, and $L_1 = (5, 5, 3, 1)$ resp. $L_2 = (5, 5, 5, 2, 1)$ with $g = 10$ for the mass-spring-mass system.

We examine all possible combinations: a parametric model (as in [123]) in combination with either our HGO (4.2) or the extended HGO (4.54), or a GP model (as in Sec. 4.2) combined with either our HGO or the extended HGO.

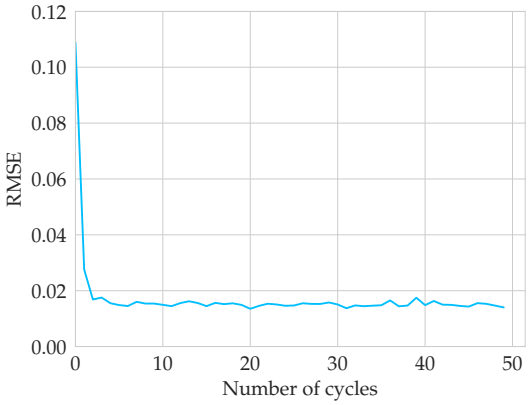
Parametric model We start with simulations of the Duffing oscillator (4.47) with sinusoidal control $u(t) = 0.4\cos(1.2t)$. We choose a parametric model:

$$\begin{aligned}\mu(x) &= \theta_1^\top \sigma(x),\\ \hat{\phi}(x) &= \theta_2^\top \sigma(x),\\ \sigma(x) &:= \begin{pmatrix} -x_1 & -x_1^3 & -x_2 \end{pmatrix},\end{aligned} \tag{4.59}$$

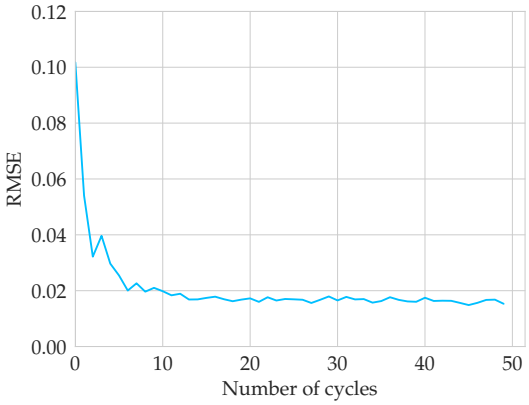
where $\sigma(x)$ is a feature vector. In general, method 1 learns a discrete model $\mu : \hat{x}(t) \mapsto \hat{x}_d(t+\Delta t)$. As discussed in Remark 6, we rather learn a residual model $\mu_{res} : (\hat{x}, u) \mapsto \hat{x}_d(t+\Delta t) - \hat{x}_d(t) - u(t)\Delta t$ instead of μ, and use $\mu(\hat{x}, u) = \mu_{res}(\hat{x}, u) + \hat{x}_d(t) + u(t)\Delta t$ for prediction. This enables formulating a linear regression problem that includes all available prior knowledge, and taking out the influence of u so that the parametric model only depends on x, as in [123]. We consider the ground truth to be $\theta_1 \simeq (-0.02996219, 0.02985621, 0.00972681)$ for method 1, which was obtained with a linear regression on the true data (without the observer). On the other hand, method 2 learns a continuous model $\hat{\phi} : \hat{x}(t) \mapsto \dot{\xi}(t)$ with a true parameter vector $\theta_2 = (-1, 1, 0.3)$. We use a linear least squares regression to estimate both parameter vectors from the corresponding datasets, obtained with the corresponding observers. We set $d = 0$ (no process noise) and ϵ to Gaussian noise of variance $\sigma_\epsilon^2 = 10^{-5}$. We run each method for ten cycles of 15 seconds for the Duffing oscillator with $\Delta t = 0.03\,\mathrm{s}$, the other parameters set as in Sec. 4.4.1.

In Figure 4.9, the state estimation RMSE, relative parameter error, and RMSE over an arbitrary test trajectory over 50 cycles are shown. As expected, all three errors decrease over time and converge up to the optimal model error and the perturbations. The final state estimation error is similar for both methods, the effect of the numerical error introduced by the Euler differentiation step in method 1 and higher noise amplification due to the extra dimension in method 2 appear to balance each other out. However, the relative error on the parameters is much lower with method 1 than with method 2, and drops much faster. This is consistent with our claim that method 1 yields a more accurate dynamics model with a given amount of samples, by incorporating less bias into the data and directly learning a discrete model. We also compare both parametric models by simulating a test trajectory with known initial state, using an RK4/5 numerical solver for both continuous models (with the Euler differentiation step for method 1, and natively in method 2). We obtain similar results as for the previous plot, though the error remains relatively high due to the chaotic nature of the considered system, which quickly accumulates numerical errors and errors on the parameters.

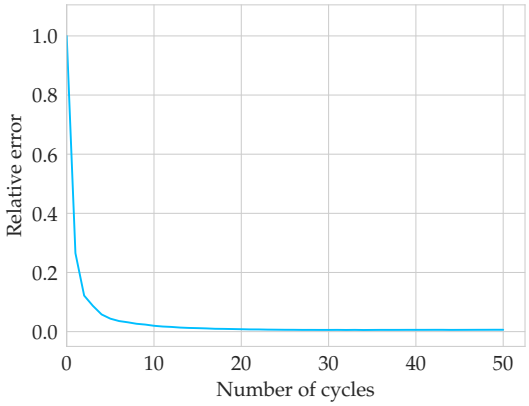
Gaussian process model We also run the same comparison with a GP model instead of a parametric model, for both the Duffing oscillator and the mass-spring-mass system presented earlier, in the same settings as in Sec. 4.4. In Figure 4.10, we compare the state estimation (top row) and the open loop prediction (bottom row) over one random trajectory of the Duffing oscillator. Though both observers show similar performance, the extended observer of method 2 (right) has higher peaking and noise amplification due to the higher dimensionality. The predictions of the dynamics model obtained with method 1 (left) also stay accurate longer than those obtained with method 2 (right). This is again consistent with our claim that a discrete model learned directly from sampled trajectories yields a more accurate dynamics model, which in turn helps to obtain a more accurate observer. The results of the mass-spring-mass system presented in Figure 4.11 are consistent with this analysis, but the



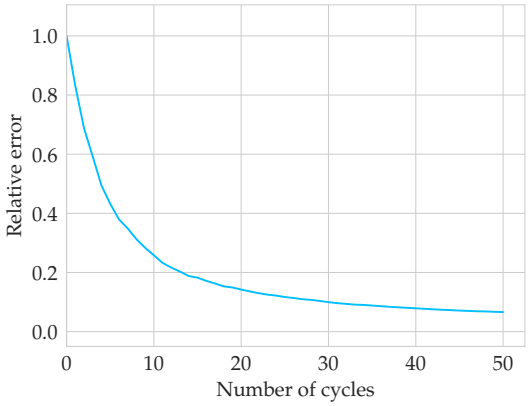
(a) State estimation error, method 1



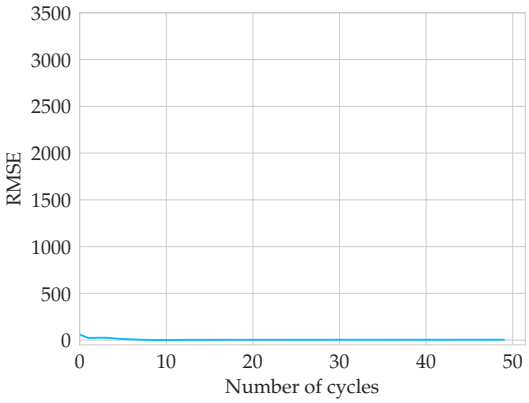
(b) State estimation error, method 2



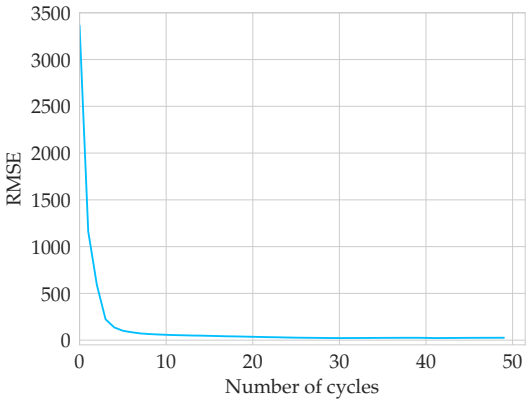
(c) Relative parameter error, method 1



(d) Relative parameter error, method 2

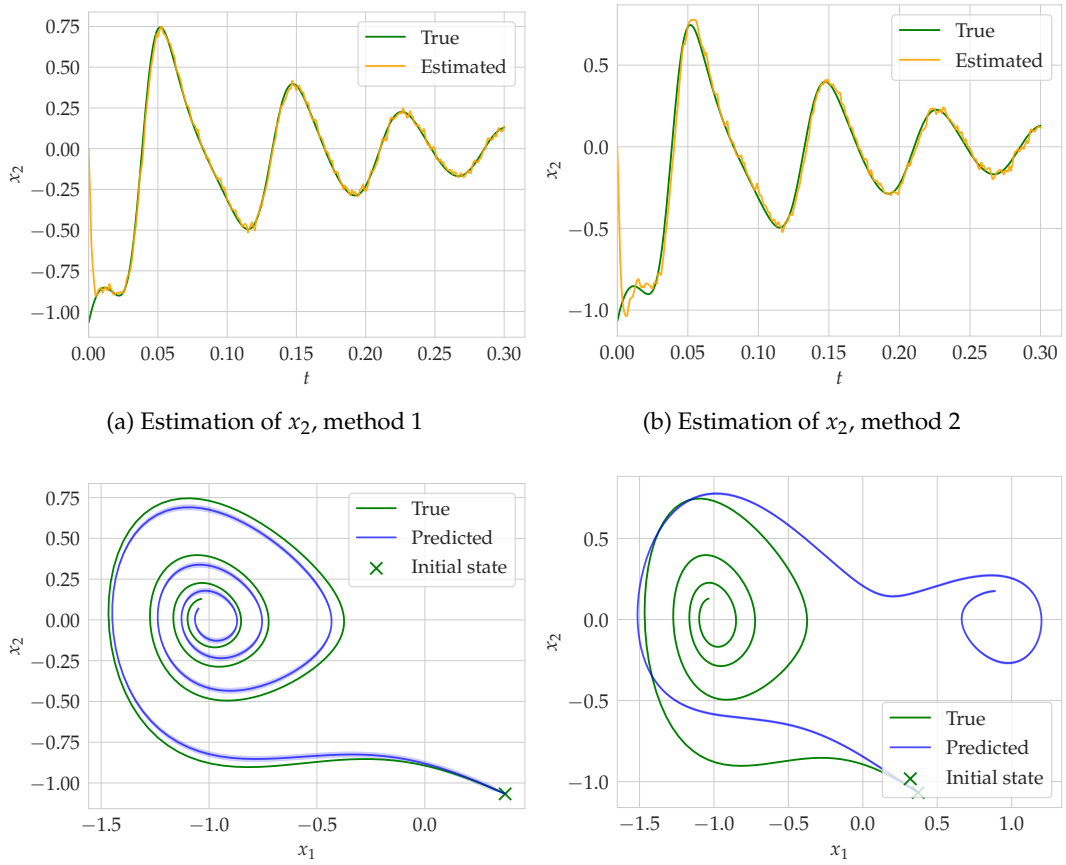


(e) RMSE over test trajectory, method 1



(f) RMSE over test trajectory, method 2

Figure 4.9: Comparison of methods 1 and 2 using a parametric model and recursive least squares to learn the dynamics of the Duffing oscillator. We plot the state estimation RMSE over the last cycle, the relative parameter error $\left|\hat{\theta} - \theta\right| / |\theta|$, where θ is the true parameter vector and $\hat{\theta}$ its current estimate, and the RMSE over an arbitrary test trajectory.



(a) Estimation of x_2, method 1

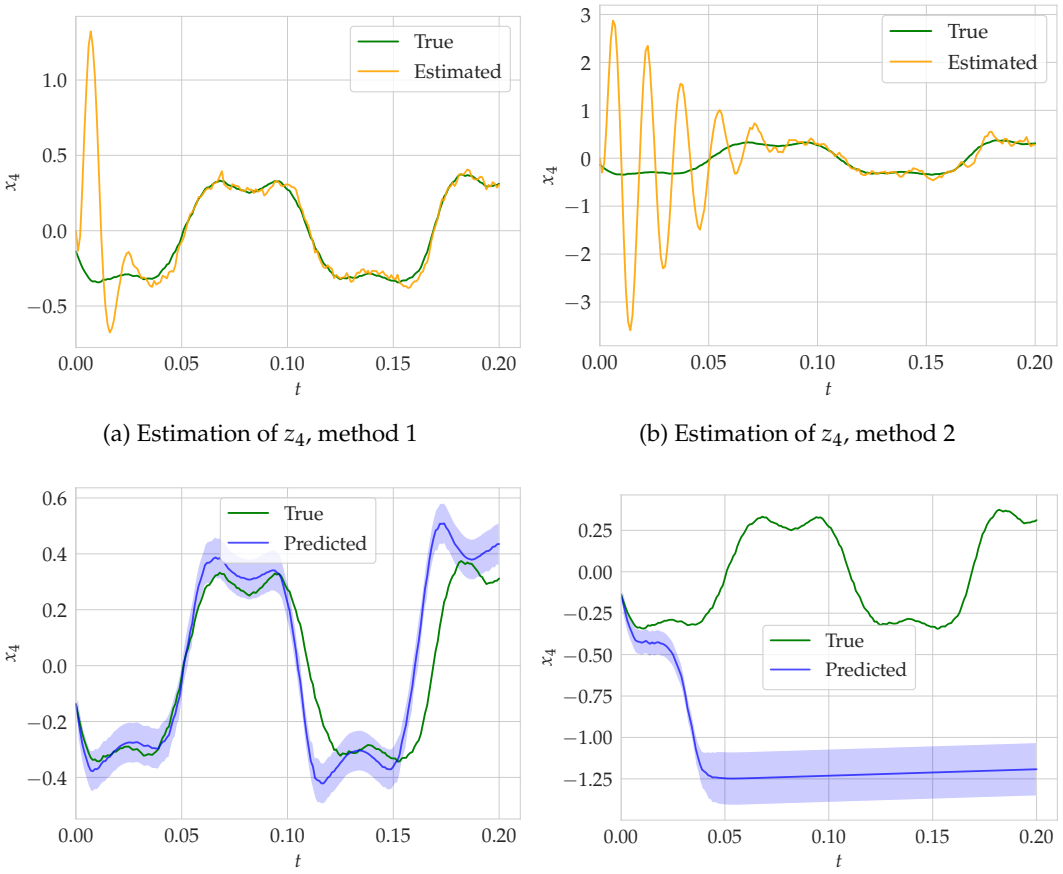
(b) Estimation of x_2, method 2

(c) Prediction of x_2 against x_1, method 1

(d) Prediction of x_2 against x_1, method 2

Figure 4.10: Comparison of methods 1 and 2 using a GP model to learn the dynamics of the Duffing oscillator: one random test trajectory with random control, after ten cycles.

difference between both methods is more important due to the higher dimensionality of the system and the more complex, input-dependent nonlinearity.



(a) Estimation of z_4, method 1

(b) Estimation of z_4, method 2

(c) Prediction of z_4, method 1

(d) Prediction of z_4, method 2

Figure 4.11: Comparison of methods 1 and 2 using a GP model to learn the dynamics of the mass-spring-mass system: one random test trajectory with random control, after ten cycles.

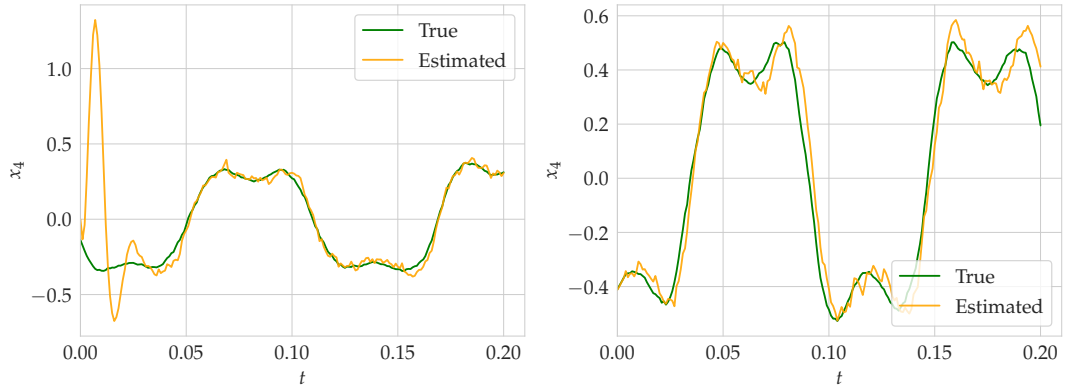
4.6 Performance improvements

The proposed method seems promising, however, its application to real-world data remains challenging. In this section, we propose two extensions for improving its performance and demonstrate them on numerical simulations.

4.6.1 Backward smoothing

Due to the asymptotic convergence of the observer, the estimation of $\hat{x}(t)$ at the end of each cycle is more accurate than at the beginning. Since model learning is time-consuming and will probably not run in real-time, it may be interesting to smooth out the trajectory estimated by the observer before updating the dynamics model. This method of going back and forth along a trajectory with an observer is referred to as back and forth nudging [127], [128]. It can be used after the measurements have been recorded to smooth out the estimated past trajectory, and even converge to the true trajectory in the absence of noise.

We propose to use a backward HGO with gains chosen appropriately, i.e., differently from the forward gains. The back and forth nudging algorithm [127], [128] runs an observer forward in time and a different observer backward in time, which in the



(a) Without smoothing

(b) With smoothing

Figure 4.12: Comparison between two estimated trajectories of the mass-spring-mass system after ten cycles, with and without smoothing.

linear case leads to two different stability conditions for the error system:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + K(y - C\hat{x}) \qquad t \in [0, T] \\ \dot{\bar{x}} &= A\bar{x} - K'(y - C\bar{x}) \qquad t \in [T, 0] \end{aligned} \tag{4.60}$$

the observers in forward and backward time respectively, leading to the stability conditions for the error system

$$\begin{aligned} A - KC &\prec 0 \\ A + K'C &\succ 0. \end{aligned} \tag{4.61}$$

We apply this idea to the previous numerical simulations. Note that due to the chaotic nature of the considered systems, the accumulation of numerical errors can lead to large changes in the obtained solution between forward and backward pass. We avoid this effect by lowering the relative solver tolerance to 10^{-8}.

We choose the gain matrix of the HGO to satisfy the given stability conditions, with the same eigenvalues (in absolute value) and the same gain for the forward and backward pass. Since the signal after the backward pass is noncausal, it may be hard to fit a causal dynamics model to this signal. Therefore, we run one forward, one backward, and again one forward pass, and use this estimated trajectory for learning, which has been smoothed but is still structurally similar to the original estimate. The results on the Duffing and the mass-spring-mass use cases are roughly the same after ten cycles as they were before smoothing, which is consistent since smoothing mainly plays a role during the first cycle, when the estimate of the initial condition is most off. However, it helps kickstart our adaptive observer by providing a better initial dataset. It also provides much smoother closed loop test trajectories estimated by the observer, since it smoothes out the initial transient. This is illustrated in Figure 4.12, where two similar estimated test trajectories are compared with and without smoothing: it is clear that the initial peaking phenomenon has been reduced in the second figure.

4.6.2 Adaptive observer gain

In the proposed framework, the dynamics model is adapted cyclically, then included in the observer. Therefore, the state estimates become more accurate after each cycle thanks to an updated dynamics model. The gain of the observer needs to be high for the initial estimates to be accurate enough to learn an initial dynamics model, so as to kickstart the joint convergence. However, as this model gets closer to the ground truth, there is no need for the gain to be so high. When possible, lowering the gain is desirable to reduce peaking and noise sensitivity. Hence, lowering the gain as the dynamics model improves would maintain the convergence of the framework while enhancing practical performance.

High-gain observers with adaptive gain have been investigated in the literature. However, the proposed adaptation laws are often nondecreasing: most works start with a low gain and increase it until the desired convergence level is reached, e.g., [129]. In our settings, the opposite type of adaptation is needed: we start with a high gain to ensure accurate state estimates even with large model errors, then wish to decrease it as the model error shrinks while the model is being learned. Recent approaches discussed in [130] search for an appropriate gain by decreasing it if the state estimation error is decreasing, and increasing it otherwise. However, information about the nonlinearity and the noise level is necessary. As opposed to these previous approaches, the adaptation law proposed in [130] directly depends on the output error of the system, and is designed to increase the gain if this error has been too large for a certain time and decrease it otherwise. Under the assumption that the model error is bounded and Lipschitz w.r.t. the measurement noise, and using only upper bounds on these constants, along with some more formal but classic assumptions, it is shown that asymptotically, the mean state estimation error is bounded by a quantity depending on some user-chosen parameters and the noise. Without model error, this result reduces to the state estimation error being input-to-state stable to the measurement noise, and with neither model error nor measurement noise, to asymptotic practical convergence. A related method is proposed in [131], with a switch between a high-gain and a Kalman mode depending on the output error over a time interval.

We combine the adaptation law proposed in [130] with our framework. The gain of the HGO g is now a time-dependent function, satisfying the ODE

$$\dot{g} = \varphi(g, \hat{y} - y) \tag{4.62}$$

with $\hat{y} = C\hat{x}$ the estimated output and φ the adaptation law defined as

$$\varphi(g, \hat{y} - y) := p_1 \left(((\hat{y} - y)^2 - p_2) g^{1-2b} + \frac{p_2}{g^{2d_x}} \right), \tag{4.63}$$

where p_1, p_2 and $b \in [0, 1/2]$ are user-defined parameters. Some tuning is necessary: the values of p_2 and b influence the threshold at which the gain stabilizes (a low p_2 and higher b are necessary to obtain a reasonably high final gain), while the combination of p_1 and b influences the rate of convergence (high p_1 for fast convergence, and low b for a more exponential profile). We set $p_1 = 4$, $p_2 = 0.002$, $b = 0.2$ and $g(0) = 12$, such that the gain decreases fast enough to see a significant difference in our experiments but slowly enough for the lower gain to be compensated by a better dynamics model. The value of the gain over time in one experiment with the mass-spring-mass system is shown in Fig. 4.13. We obtain satisfying results: the dynamics model converges to a good estimate of f, while the final observer has a lower gain and therefore better

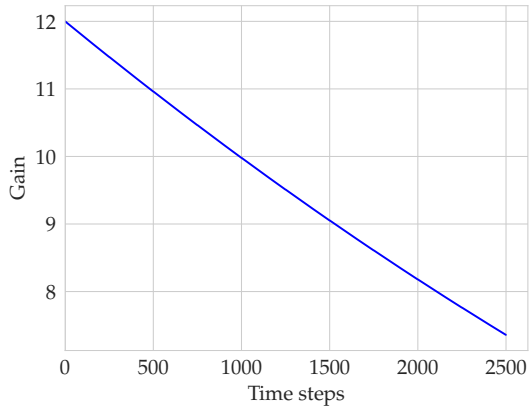
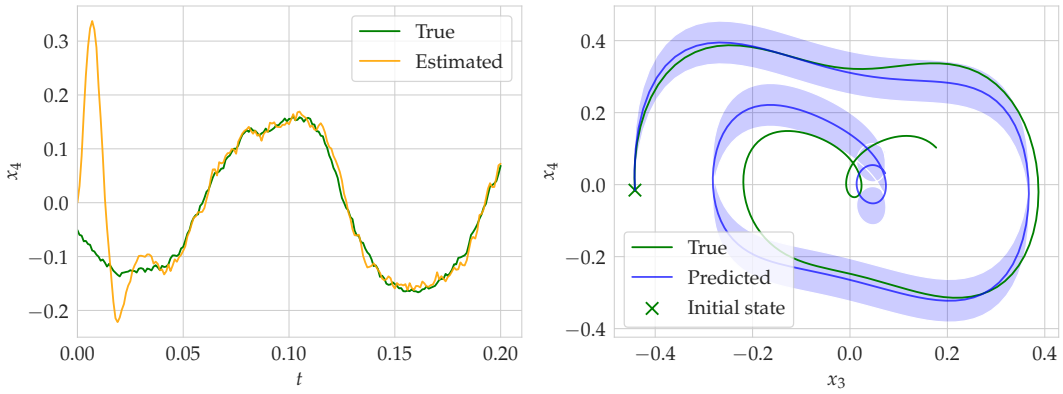


Figure 4.13: Evolution of the adapted gain over time, for the mass-spring-mass system.



(a) Closed loop test trajectory

(b) Open loop test trajectory

Figure 4.14: Test trajectories of the mass-spring-mass system after ten cycles, with adapted gain. To be compared with Fig. 4.8–4.7.

performance in the presence of measurement noise. This is demonstrated in Fig. 4.14, which depicts one open loop and one closed loop test trajectory. Compared to Fig. 4.8–4.7 with fixed gain $g = 10$, the prediction accuracy in open loop is similar while the state estimate is less noisy due to the lower final gain. The quality of the obtained observer and dynamics model then depends on the user's choices and on how fast the learning procedure takes place. Overall, with this extension of the proposed framework, the user can start from a rather high gain and no system knowledge, and obtain an accurate dynamics model and an observer with relatively low gain.

4.7 Extension: Ansys Fluent use case

We now apply the proposed method for joint state and dynamics estimation to a more complex use case constructed using Ansys software. It consists of a fluid dynamics system for which a prior model is known, but inaccurate due to changes in the system. We aim at correcting the dynamics model using partial observations.

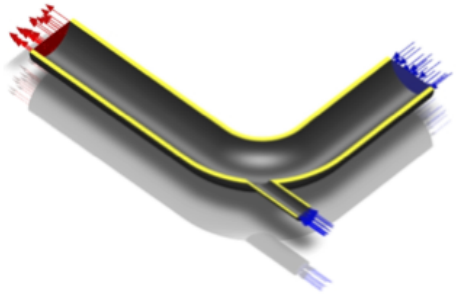


Figure 4.15: Diagram of the elbow conduct in the Ansys Fluent use case. The temperature sensor on the surface of the conduct is located towards the end, after the fluids have mixed.

We simulate the mixture of hot steam and air in an elbow-shaped conduct with two inputs and one output, as illustrated in Figure 4.15. This fluid dynamics simulation is run in Ansys Fluent, and the temperature evolution in the solid conduct while the two fluids mix is computed over a field of about 2×10^4 points in time for 10^4 seconds. We generate two datasets: one with a heat transfer coefficient of $5W.m^{-2}.K^{-1}$ for the conduct, corresponding to the known system over which the prior dynamics model is learned, and a modified dataset with a heat transfer coefficient of $8W.m^{-2}.K^{-1}$, corresponding to the true physical system after some change in the dynamics, e.g., due to wear. Then, the input-output behavior of the original dataset is learned to build a prior model of the system, representing our initial knowledge of it. In the last step, we refine this prior using partial measurements of the "true" system, i.e., the modified dataset, which represents the physical system, slightly different from the theoretical one.

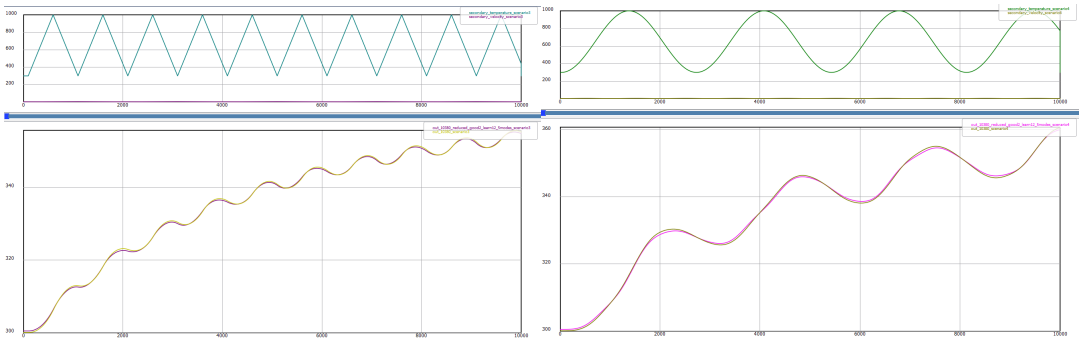
4.7.1 Reduced order modeling

We now describe how we obtain this prior model, first by learning a reduced order model (ROM) with Ansys DynaROM, then by fitting a GP.

To model the input-output behavior of the original system, a rectangular singular value decomposition (SVD) of the temperature field on the surface of the solid conduct is computed. The evolution of the first five modes is recorded:

$$T_{field} \simeq U\Sigma V^* = UM, \tag{4.64}$$

where T_{field} is the whole spatiotemporal temperature field over $N = 2 \times 10^4$ points and $n = 10^4$ time steps, U is an n by 5 matrix containing the value of the five SVD modes at each time step, and M is a 5 by N matrix containing the SVD coefficients of each point in the temperature field. The five SVD modes in U serve as state, so that $x(t)$ at time t is a linear interpolation between the corresponding lines in U, while the fluid temperature and velocity at the secondary outlet (small tube on the left) serve as two inputs $u(t)$ to the system. The data consists of four scenarios with different control strategies: step, sinusoidal at two different frequencies, and triangle inputs. Note that this use case and these scenarios are used at Ansys as demonstrators for DynaROM. In all subsequent graphs, we use $C^\top x(t)$ as ground truth, where C is the vector of SVD coefficients of the considered point in the temperature field, and $x(t)$ is the vector of SVD modes given by the Ansys SVD tool at the given time step. Hence, the trajectories marked as "True" are not directly the measurements, but the



(a) Scenario 3 (b) Scenario 4

Figure 4.16: Test scenarios with the original dataset (before modification): predictions of the DynaROM model, mapped to the temperature at the sensor location via its SVD coefficients. Input above, output below, with the true trajectory in mustard and the prediction in pink.

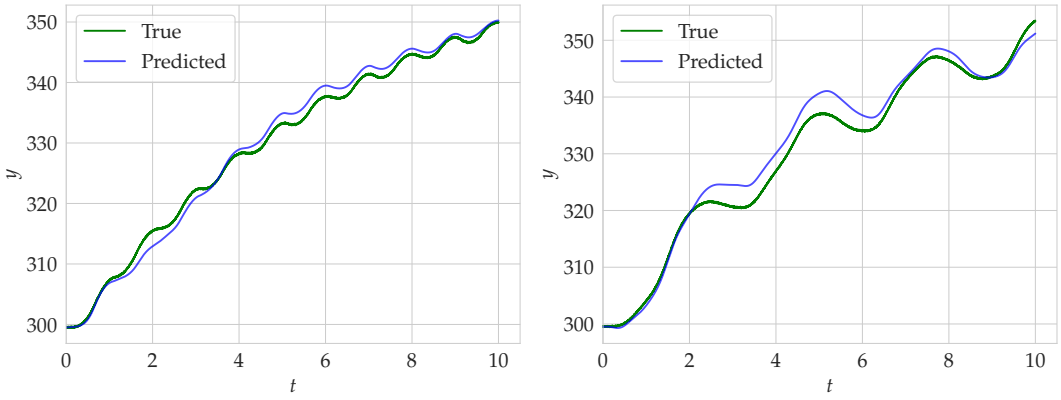
reconstitution of this measurement given the SVD decomposition, which is very close to the true measurement in this example when using 5 modes.

The prior model is learned on the first two scenarios, then tested on the two next ones. We learn a first model with Ansys DynaROM, which iteratively and automatically determines how many hidden states are necessary to explain the data. Here, we consider 5 SVD modes after some trial and error with DynaROM, showing that $d_x = 5$ is enough to obtain satisfying accuracy. The trained ROM fits the original training and test scenarios with high accuracy and rather short training time, thanks to the efficient implementation including automatic hyperparameter tuning features. Its performance is illustrated in Fig. 4.16 on the test scenarios. This gives us confidence in the design of the experiment: with $d_x = 5$ and the chosen training and test scenarios, it should be possible to fit the original dataset and obtain a prior model on the whole state.

For ease of implementation, we do not directly use the Ansys ROM as our prior model. Instead, once DynaROM has provided us with a proof of concept and validation of the settings, we fit a GP model to the original dataset to obtain a prior GP model, which we can then directly refine given measurements from the modified dataset. Since 20^4 data samples are available for learning, which is intractable for vanilla GPs, we use the sparse approximation "VarDTC" (see Sec. 3.1.1 and [57] for an overview) and only keep 500 points in memory. We also optimize the hyperparameters of the squared exponential kernel offline on the first 2000 data points without sparsification, which yields a scaling factor of 10 and lengthscales $(150, 80, 90, 90, 90, 150, 150)$. The obtained model, further denoted f_0, maps an SVD state $(x(t), u(t))$ of dimension $d_x = 5$ and an input of dimension $d_u = 2$ to $x(t + \Delta t)$. Its performance is illustrated in Fig. 4.17, where it is shown that the prior GP model can accurately predict the test scenarios of the original dataset.

4.7.2 Correcting the ROM

Once the prior model has been trained, we apply our framework to refine it using partial measurements from the modified system, instead of whole state data (full SVD modes). The measurement is the temperature at one point of the conduct, corrupted with Gaussian noise of variance 0.0026, to mimic a single sensor measuring



(a) Scenario 3

(b) Scenario 4

Figure 4.17: Test scenarios with the original dataset (before modification): predictions of the prior GP model, mapped to the temperature at the sensor location via its SVD coefficients.

the temperature field at a specific point (in orange in Figure 4.15). This serves as the partial, noisy output of the system: we have $y = Cx + \epsilon$, where C is the vector of SVD coefficients at this specific point in the temperature field (we assume the SVD decomposition computed for the original system is still valid for the modified one). We can now plug this y into our joint state and dynamics estimation method, estimate the state trajectory $\hat{x}$, and update the discrete GP model from it. However, the prior model learned from the original dataset is not in the observable canonical form. We cannot transform the coordinate system as for the mass-spring-mass system, as it will be necessary to recover the temperature at any point from the predicted value of the SVD modes using the SVD decomposition. Therefore, we implement an extended Kalman filter (EKF) instead of an HGO, using the prior model as the dynamics to be linearized in the EKF and relying on it to obtain reasonably accurate state estimates. We then construct and learn a GP model as previously described, learning the residuals[2] of the prior model. This does not exactly fit into our theoretical guarantees, but demonstrates the performance of the proposed method in a large-scale, practical setting.

We run two cycles of state estimation followed by a GP update, over the two training scenarios. The initial conditions of the scenarios are considered known, since they were chosen by the user to train the prior model on simulation data. Starting with a good initial guess eases the task of the observer, yielding more accurate training trajectories for the dynamics model. With a reasonably accurate prior model and after tuning the EKF, we obtain convincing results: the state trajectories estimated by the EKF are accurate enough to improve the dynamics model iteratively from the partial, noisy measurement data. The results are illustrated in Fig. 4.18, where the temperature at the sensor in the test scenarios is shown. The refined model, trained using the state trajectories estimated from the sensor measurements, can predict the output much better than the prior model (prediction RMSE reduced by 81% for scenario 3, 58% for scenario 4). The same predictions at another, unobserved location in the temperature field are presented in Fig. 4.19. The refined model also yields a reduction in RMSE of 56% for scenario 3, 46% for scenario 4.

[2]Recall that learning the residuals of a prior model means learning the mapping from the inputs to the difference between the output of the prior and the observed output.

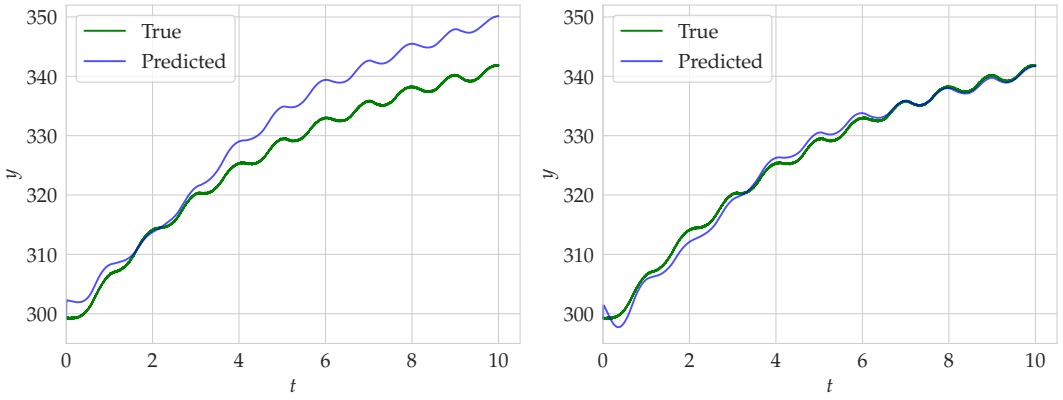
It is also possible to start with a different value of $\hat{x}(0)$, e.g., zero. However, this leads to a large transient in the estimated trajectories, which needs to be cut out from the training data. With only limited scenarios to learn from, this loss of information degrades the performance of the obtained dynamics model. As an example, we show in Fig. 4.20 the prediction at the sensor location and at the test point with $\hat{x}(0) = 0$, where the first hundred time steps of the estimated trajectories have been cut out of the training data. The prior model remains the same. The accuracy still improves compared to the prior, but the performance degrades compared to Fig. 4.18–4.19, most of all at the beginning of the trajectory. Nonetheless, workarounds for this problem can be envisioned, such as starting from a reasonable initial guess or smoothing the trajectory estimated by the EKF. For example, the equivalent in this context of adding a backward pass to smooth out the trajectory estimated by the HGO is to run a Rauch-Tung-Striebel smoother over the trajectory estimated by the EKF [132, Sec. 8.2]. This smoother somewhat reduces the transient of the EKF, but it is too dependent on the EKF estimates to completely correct it in a back and forth nudging fashion.

4.7.3 Discussion on the EKF extension

The previous results demonstrate that the proposed cyclic framework can be extended even when our assumptions are not satisfied, provided a reasonably accurate prior model and initial state estimates for the first cycle. This use case is meant to mimic the problem of hybrid digital twins: a high-fidelity simulator (Fluent) is used to generate the original dataset and learn a prior model of the system (either a ROM, a prior GP, or some other type). Then, noisy and partial physical measurements (temperature sensor on the conduct) are used to refine the model with a GP to account for changes in the dynamics (modified dataset), such as wear or parameter deviation over time. It serves as proof of concept that the proposed framework can function in industrial settings to fine-tune digital twins using partial physical measurements. However, no convergence guarantees are provided for this extension using the EKF, and hyperparameter tuning can be challenging: the EKF gains, the GP hyperparameters, and the training data need to be well chosen.

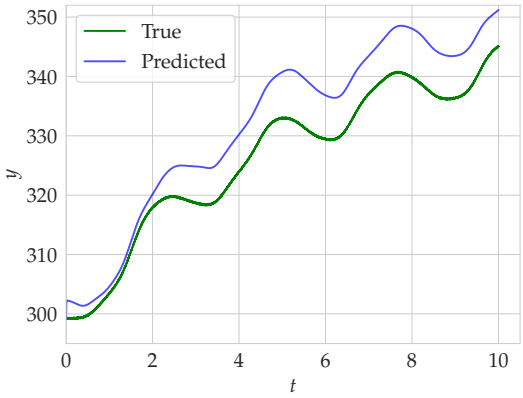
4.8 Discussion

In this chapter, we propose a framework for joint state and dynamics estimation of systems in the observable canonical form. Though the method shows promising results, it has some limitations that require further investigation. On the observer side, Theorems 3 and 4, like all theoretical guarantees for HGOs, only hold for high enough gain. However, using a large gain can be prohibitive in practice, mostly in high dimensions or with high measurement noise, as HGOs suffer from peaking and noise amplification. This can be seen in Figure 4.7, where peaking is present and the measurement noise is already visible though it was rather low in the simulations ($\sigma_\epsilon = 10^{-4}$), making it difficult to deal with much higher noise levels. These effects can be mitigated by changing the gain **Khalil_highgain_meas_noise_switche\bar {d}ain** or using a cascade of HGOs [120], [133]. In future work, these could be combined with our method to allow for higher noise levels. We are also limited by the observable canonical form: many systems can be transformed into this form without knowledge of the dynamics, as in Sec. 4.4, but the transformation back into the original coordinates remains unknown. Using an EKF instead of the HGO as in the last section circumvents some of these issues, however, EKFs are notably hard to tune and not very robust to model

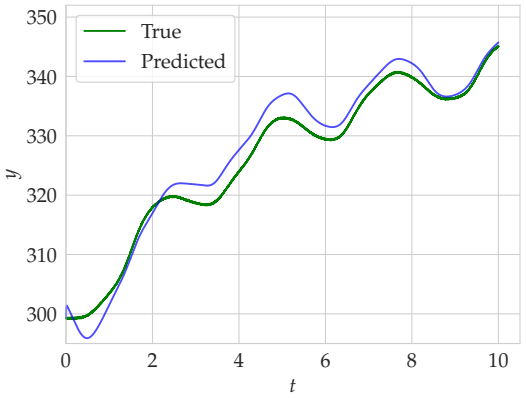


(a) Scenario 3, prior model

(b) Scenario 3, after learning



(c) Scenario 4, prior model

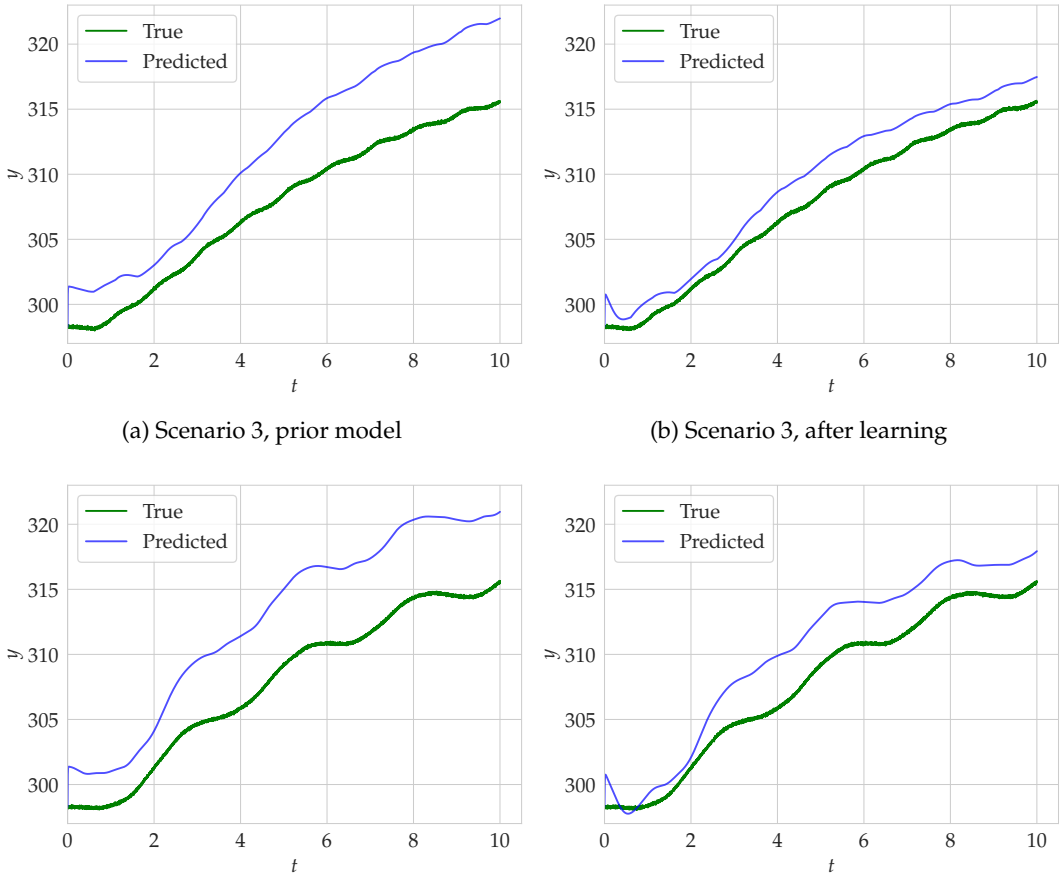


(d) Scenario 4, after learning

Figure 4.18: Test scenarios before and after refining the dynamics model for the Ansys Fluent use case. The noisy measurement and predicted output (open loop) are shown; the updated GP model learns the residuals of the prior model.

errors, such that a reasonably accurate prior model of the dynamics is needed. In the next chapter, we extend this methodology to more general classes of systems, at the cost of the theoretical guarantees provided by HGOs.

On the learning side, we assume $\hat{f}$ is Lipschitz continuous w.r.t. the training data. This is the case for GPs with fixed hyperparameters. However, as soon as a nonconvex optimization procedure is involved, e.g., for online hyperparameter tuning or for training a neural network instead of a GP, this assumption is not satisfied as the obtained solution can change non smoothly if the training data changes. In this case, Theorems 3 and 4 do not hold. In practice, such tools can often still be used while maintaining performance. Another limitation of GPs is the dimensionality: as the computation of the posterior scales at $O(N^3)$, the GP model can only deal with up to about 10^4 data samples. The sliding window of length N used in (4.7) deals with this issue but leads to the loss of possibly useful data. Other methods such as sparse GP approximations [57] can be more efficient in practice, as shown in the Fluent use case, but the theoretical guarantees do not extend to these approximations based on nonconvex optimization.



(a) Scenario 3, prior model

(b) Scenario 3, after learning

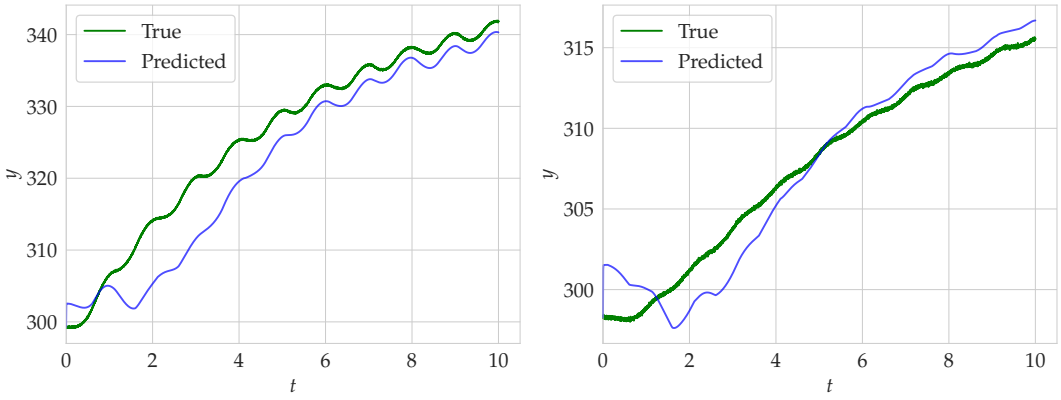
(c) Scenario 4, prior model

(d) Scenario 4, after learning

Figure 4.19: Test scenarios before and after refining the dynamics model for the Ansys Fluent use case. The true and predicted output (open loop) at another, unobserved point in the temperature field are shown; the updated GP model learns the residuals of the prior model.

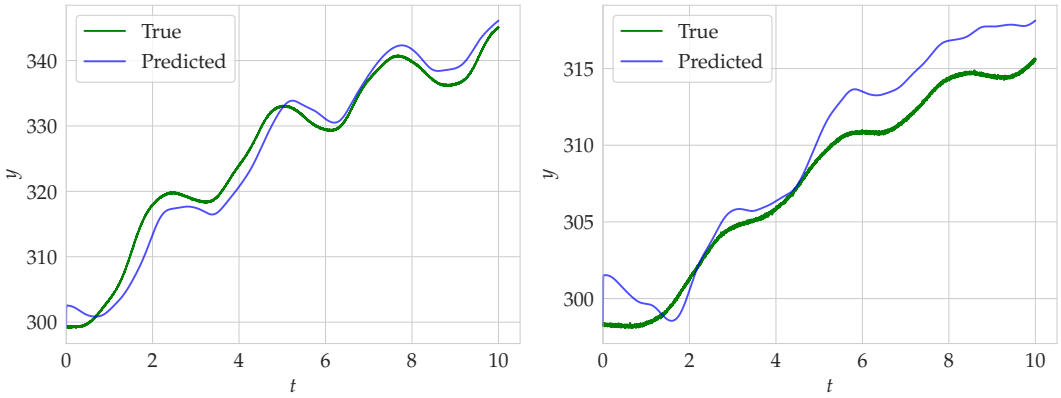
4.9 Conclusion and outlook

Due to the imperfect state data provided by most physical platforms, joint estimation of state and dynamics is at the core of dynamics model learning from experimental data. This is a challenging problem in general: few approaches exist, even fewer provide convergence guarantees. In this chapter, we propose a framework for joint state estimation and dynamics learning of nonlinear systems in the observable canonical form. A high-gain observer estimates the state trajectory, which is used for learning the nonlinearity with a non-parametric Gaussian process model. Practical and asymptotic convergence of both the state and dynamics estimation can be guaranteed, so that after convergence, the observer and the dynamics model can be used for further control tasks. Simultaneous model learning and improved state estimation are demonstrated on two academic examples. We then discuss a quantitative comparison to the most related existing work, which inspired our method. We also propose two methodological improvements: running the observer in a back and forth nudging manner to smooth out the data before each learning cycle, and adapting the observer gain as the dynamics model improves. An extension of the method is investigated, where replacing the HGO with an EKF circumvents the assumption of observable



(a) Scenario 3, sensor location

(b) Scenario 3, unseen location



(c) Scenario 4, sensor location

(d) Scenario 4, unseen location

Figure 4.20: Test scenarios after refining the dynamics model for the Ansys Fluent use case, at the sensor location (left) and at an unobserved point (right). The updated GP has improved compared to the prior model, but the performance is degraded due to the inaccurate initial guess $\hat{x}(0)$.

canonical form so that a general prior model can be refined with measurement data. This is demonstrated on a larger scale use case provided by Ansys.

We now look to extend the methodology further, by leveraging deep learning and automatic differentiation techniques to learn dynamics models from partial observations with varying degrees of prior knowledge.

Chapter 5

Learning neural ODEs from partial observations with recognition models

Résumé L'identification de systèmes dynamiques à partir de données expérimentales est une tâche particulièrement difficile, entre autres parce que les données disponibles sont rarement denses et représentatives. Les connaissances préalables sur le modèle améliorent généralement l'efficacité de l'identification, mais l'étendue de ces connaissances varie en fonction de l'application, et des modèles spécifiques à chaque cas d'usage sont souvent nécessaires. Les équations différentielles ordinaires neuronales donnent un cadre flexible pour l'identification de systèmes et peuvent incorporer un large spectre de connaissances physiques, donnant une interprétabilité physique à l'espace latent résultant. Dans le cas d'observations partielles, cependant, les données ne peuvent pas être directement mises en correspondance avec l'état latent de l'équation différentielle ordinaire. Nous proposons donc des modèles de reconnaissance, notamment inspirés de la théorie des observateurs non linéaires, pour relier les observations partielles à l'état latent. Nous illustrons la performance de l'approche proposée sur des simulations numériques et sur des données expérimentales provenant d'un exosquelette robotique.

Abstract Identifying dynamical systems from experimental data is a notably difficult task, in particular due to the difficulty of collecting dense and representative data. Prior knowledge generally improves the model's sample efficiency, but the extent of this knowledge varies with the application, and customized models are often needed. Neural ordinary differential equations can be written as a flexible framework for system identification and can incorporate a broad spectrum of physical insight, giving physical interpretability to the resulting latent space. In the case of partial observations, however, the data points cannot directly be mapped to the latent state of the ODE. Hence, we propose to design recognition models, in particular inspired by nonlinear observer theory, to link the partial observations to the latent state. We demonstrate the performance of the proposed approach on numerical simulations and on an experimental dataset from a robotic exoskeleton.

Parts of this chapter are published in the Transactions of Machine Learning Research under the title *Recognition Models to Learn Dynamics from Partial Observations with Neural ODEs* [50].

5.1 Introduction

The dynamic behavior of complex physical systems often follows a certain structure. Mathematically, this structure is captured by differential equations, e.g., the laws of physics. However, even an accurate model cannot account for all aspects of a physical phenomenon, and physical parameters can only be measured with uncertainty. Data-driven methods aim to enhance our predictive capabilities for complex systems based on experimental data.

We focus on dynamical systems and design an end-to-end method for learning them from experimental data. We investigate State-Space Models (SSMs), which are common in system theory, as many modern control synthesis methods build on them and the states are often amenable to physical interpretation. For many systems of interest, some degree of prior knowledge is available. It is desirable to include this knowledge in the SSM. To this end, we consider neural ordinary differential equations (NODEs), which were introduced by [32] and have since sparked significant interest, e.g., [39], [104], [134], [135]. They aim to approximate a vector field that generates the observed data following a continuous-time ODE with a neural network. Their formulation is general enough to avoid needing a new design for each new system, but can also enforce a wide range of physical insight, allowing for a meaningful and interpretable model. Specific approaches have been proposed to include different priors, which we briefly recall; we present a unified view and include them in the proposed end-to-end framework.

Learning an SSM satisfying these priors amounts to learning the dynamics in a specific set of coordinates. However, experimental data is typically only partial, as not all of these coordinates, or *states*, are measured. This is a common problem in machine learning, where the existence of an underlying state that can explain the data is often assumed. In the systems and control community, estimating this underlying state from partial observations is known as state estimation or observer design. An observer is an algorithm that estimates the full latent state given a history of partial measurements and control inputs [42], [43]. While observer design provides the theoretical framework for latent state estimation, such as convergence guarantees of the estimate to the true state, it has not received much attention in the machine learning community. Hence, we propose to leverage concepts from nonlinear observer design to learn NODEs with physical priors from partial observations.

We design so-called *recognition models* to map the partial observations to the latent state. We discuss several approaches, in particular based on a type of nonlinear observers called Kazantzis-Kravaris/Luenberger (KKL) observers [46], [47]. We show that the KKL-based recognition models perform well and have desirable properties, e.g., a given size for the internal state. Such recognition models can then be embedded in the NODE formulation or any other optimization-based system identification algorithm. Our main contributions can be summarized as follows:

- We discuss related work in more detail in Sec. 5.2;
- In Sec. 5.3, we formulate structured NODEs as a flexible framework for learning dynamics from partial observations, which enables enforcing a broad spectrum of physical knowledge;
- We introduce recognition models to link the observations and the underlying state of the NODE in Sec. 5.4, then propose several forms based on nonlinear observer design;
- We investigate how a wide spectrum of prior knowledge can be included in the NODE formulation in Sec. 5.5, and position existing works on this spectrum;
- We compare the proposed recognition models in a simulation benchmark in Sec. 5.6.2 and on a numerical example in Sec. 5.6.3;
- We then apply the proposed approach to an experimental dataset obtained on a robotic exoskeleton in Sec. 5.6.4, illustrating the possibility of learning a physically sound model of complex dynamics from real-world data.

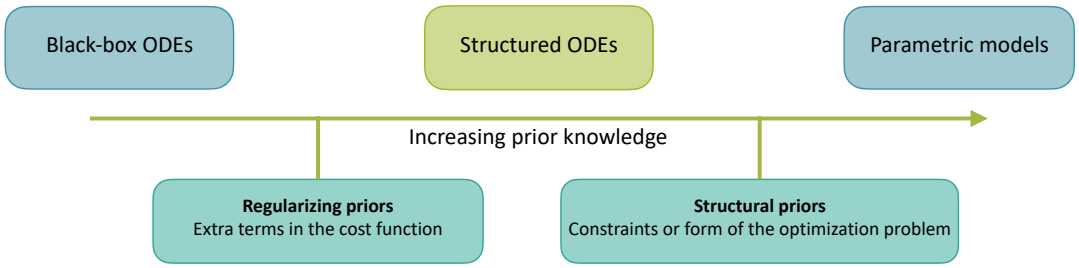


Figure 5.1: SSMs can include a broad spectrum of physical knowledge. On the left, purely data-based formulations such as latent NODEs are general but tend to violate physical principles and have trouble generalizing. On the right, parametric models can be identified from data: they extrapolate well but are system-specific and require expert knowledge. One can bridge this gap by including the available physical knowledge in an NODE formulation (5.2), in particular "regularizing" priors (extra terms in the cost function) or "structural" priors (constraints or form of the optimization problem).

Combining these yields an end-to-end framework for learning physical systems from partial observations. Finally, we discuss the limitations of the proposed approach and possible future work in Sec. 5.7.

5.2 Related work

The proposed method is based on two research areas: nonlinear observer design and machine learning for dynamical systems. We give an overview of the main trends and the most related methods on these topics.

5.2.1 System theory

In system theory, many subfields are concerned with the study of dynamical systems from experimental data.

System identification

The area of system identification aims at finding a possible dynamics model given a finite amount of partial measurements [16], [17], [136]. For linear systems, a suitable set of system matrices can be identified using subspace methods [15]. For nonlinear systems, most state-of-the-art techniques aim at estimating the variables of a given parametric model using Bayesian parameter estimation [21] or optimization-based methods [18]–[20], or a decomposition of its dynamics on a suitable basis of functions [23]. These classical methods tend to be system-specific: they require expert knowledge to construct a parametric model or precisely pick the hypothesis class in which it will be approximated. NODEs are a general tool for system identification in case no parametric model is available, in which a broad range of physical knowledge can be included by adapting the formulation.

Observer design

When identifying a state-space model from partial observations, the unknown latent state must be estimated. This is the objective of state observers or estimators, which

infer the state from observations by designing an auxiliary system driven by the measurement (see [42], [43] for an overview). Observers often assume an accurate dynamics model, but designs that can deal with imperfect models are also available. In that case, the unknown parts of the dynamics can be overridden through high-gain or sliding-mode designs to enable convergence [49], [125]. Otherwise, the unknown parameters can be seen as extra states with constant dynamics, and extended state observers can be designed, such that the estimated state and parameters converge asymptotically [137]. Some concepts from observer theory can be leveraged to improve upon existing approaches for learning dynamics from partial observations, which require estimating the unknown latent state.

5.2.2 Learning dynamical systems

Learning dynamical systems from data is also investigated in machine learning [14], [35]. We focus on settings considered realistic in system identification, i.e., methods that allow for control and partial observations, and can ensure certain physical properties.

Physics-aware models

The dynamics models obtained from machine learning often struggle with generalization and do not satisfy important physical principles such as conservation laws. Therefore, there have been efforts to bring together machine learning and first principles to learn physics-aware models; see [4], [36], [37] for an overview of these efforts in deep learning. In general, there are two takes on including physical knowledge in data-driven models, as illustrated in Fig. 5.1. On the one hand, "regularizing" priors can be included by adding terms to the cost function to penalize certain aspects. The most common case is when a prior model of the system is available from first principles. This is investigated in [138], [139] for full state observations; more details are available in Sec. 5.5.1. On the other hand, structural properties such as conservation laws can be enforced by constraints or a specific form of the optimization problem. This improves the performance and interpretability of the model; see Sec. 5.5.2 for a closer examination. However, little previous work on NODEs assumes partial and noisy measurements of system trajectories.

There exist various other methods to learn physics-aware dynamics models, such as Bayesian approaches, in which prior knowledge can be enforced in the form of the kernel [140], by structural constraints [73], [74], [76], or by estimating the variables of a parametric model [21]. In this work, we focus on NODEs, which leverage the predictive power of deep learning while enforcing a broad range of physical knowledge in the problem formulation. Note that this distinction between "regularizing" and "structural" prior knowledge can also be applied to other approaches: for example, constructing task-specific kernels for Gaussian process models can be seen as a soft form of prior knowledge, while Bayesian parameter estimation inside a parametric model [21] is a form of structural constraint.

Partial observations

Most NODE frameworks for dynamical systems assume full state measurements. Partial observations greatly increase the complexity of the problem: the latent state is unknown, leading to a large number of possible state-space representations. In this case, the question of linking the observations with the latent state needs to

be tackled. In Bayesian approaches, the distribution over the initial state can be directly conditioned on the first observations then approximated by a so-called recognition model [65], [81], [82]. Such an approach has also been used for Bayesian extensions of NODEs, where the NODE describes the dynamics of the latent state while the distribution of the initial latent variable given the observations and vice versa are approximated by encoder and decoder networks [141], [142]. The encoder network, which links observations to latent state by a deterministic mapping or by approximating the conditional distribution, can also be a Recurrent Neural Network (RNN) [105], [134], [143], [144] or an autoencoder [29], [104]. The particular case in which the latent ODE is linear and evolves according to the Koopman operator (that can be jointly approximated) is investigated in [145], [146]. In general, little insight into the desired latent representation is provided. This leads to difficulties for the obtained models to generalize, but also with their interpretability: often in a control environment, the states should have a physical meaning. Therefore, we propose to learn a recognition model that maps the observations to the latent state, while enforcing physical knowledge in the latent space.

5.3 Problem statement

Consider a general continuous-time nonlinear system

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \qquad & y(t) &= h(x(t), u(t)) + \epsilon(t) \\ x(0) &= x_0 \end{aligned} \tag{5.1}$$

where $x(t) \in \mathbb{R}^{d_x}$ is the state, $u(t) \in \mathbb{R}^{d_u}$ is the control input, $y(t) \in \mathbb{R}^{d_y}$ is the measured output, and f, h are the true dynamics and measurement functions, assumed continuously differentiable. We denote $\dot{x}(t)$ the derivative of x w.r.t. time t, and generally omit the time dependency. We only have access to partial measurements y corrupted by noise ϵ, the control input u, and the measurement function h: the dynamics f and the state x are unknown. We assume that the solutions to (5.1) are well-defined and aim to estimate f.

The objective of NODEs [32] is to learn a vector field that generates the data through an ODE, possibly up to an input and output transformation; see [33] for an overview. This type of model has recently sparked widespread interest: applications ranging from the analysis of complex materials [104] to neural activity [143] or climate factors [147] have been proposed, and their controllability and approximation properties are under investigation [39]. While the interpretation of this formulation for general machine learning tasks remains open, it is very natural for SSMs: learning the NODE boils down to approximating the dynamics with a neural network.

However, there is no unifying framework for applying NODEs to dynamical systems in realistic settings, i.e., with partial and noisy trajectory data, with a control input, and using all available physical knowledge.

Assume we have access to N measured trajectories indexed by j, denoted $\underline{y}^j$ and sampled at times t_i, $i \in \{0, ..., n-1\}$. We approximate the true dynamics f with a neural network f_θ of weights θ. If the initial conditions x_0^j are known, learning f_θ can

be formulated as the following optimization problem:

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2d_y nN} \sum_{j=1}^{N} \sum_{i=0}^{n-1} \left| y^j(t_i) - \underline{y}^j(t_i) \right|^2 := L(\theta) \\ s.t. \quad & \dot{x}^j = f_\theta(x^j, u^j) \qquad y^j = h(x^j, u^j) \\ & x^j(0) = x_0^j, \end{aligned} \tag{5.2}$$

where the constraint is valid for all $j \in \{1, ..., N\}$.

Solving NODEs Optimizing θ to find a local optimum of (5.2) requires computing the gradient of $L(\theta)$, for which several methods have been proposed. The forward sensitivity method consists in computing $\frac{\partial L}{\partial \theta}(\theta)$ explicitly, in which the derivative of the flow $\Lambda(t_i) := \frac{\partial x}{\partial \theta}(t_i)$ at each time step intervenes. This sensitivity is the solution to another ODE, which depends on the gradient of f_θ with respect to x and θ. By solving the ODE on x and Λ simultaneously in forward time, one can compute the gradient of $L(\theta)$. However, this method is computationally demanding, as it requires solving the ODE on the $d_x \times d_\theta$ matrix Λ for all t_i at each iteration of the optimization. It is known in the computational physics community [148], [149] that the adjoint sensitivity method is a computationally more efficient alternative to obtain the gradient of a cost function depending on the state of a parametrized ODE when L is scalar and d_θ is large. This method introduces continuous-time Lagrange multipliers to formulate the dual problem of (5.2). Thus, instead of simulating the ODE on the large sensitivity matrix Λ, it solves sequentially smaller ODEs and integral equations in backward time. Details on these methods are provided in [18], [33], [148]. We opt for automatic differentiation through the numerical solver (`torchdiffeq` by [32]). This is faster than the two previous methods since no other ODEs need to be solved, but has higher memory requirements due to the potentially large computational graph. It is also exact since no approximations are made, but dependent on the particular solver and options.

Problem (5.2) is not well-posed: for a given state trajectory, there exist several state-space representations f that can generate the data. This is known as the unidentifiability problem [150]. The key problems to obtain meaningful solutions to (5.2) are:

(i) enforcing physical knowledge to learn a state-space representation that not only explains the data, but is also physically meaningful;

(ii) dealing with partial observations, i.e., unknown latent state and in particular unknown x_0.

Problem (i) has been addressed in the literature for some particular cases, as presented in Sec. 5.5, by including the available physical knowledge in the form of "regularizing" or "structural" priors (see Fig. 5.1 and related work in Sec. 5.2.2). We denote the general approach of adapting (5.2) to build physics-aware NODEs as *structured NODEs*, and apply it to examples with varying prior knowledge.

Problem (ii) remains largely open. For fixed f_θ and u, each prediction made during training for a given measured trajectory is determined by the corresponding x_0; hence, estimating this initial state is critical. We tackle (ii) by designing recognition models in Sec. 5.4. We combine them with structured NODEs to simultaneously address (i) and (ii). This yields an end-to-end framework for system identification based on physical knowledge and partial, noisy observations, a common setting in practice;

we apply it to several examples in Sec. 5.6. While some of the components needed for this framework exist in the literature, e.g., on building physics-aware NODEs (Sec. 5.2.2), the vision of recognition models, the presentation in a unified framework and the application to relevant practical cases are novel.

Remark 9. *The considered setting, where h in (5.2) is known, is generally regarded as realistic in system identification, since y is measured by sensors, and u is chosen by the user, who often knows which part of the state is being measured. However, if that is not the case, it is possible to train an output network h_θ jointly with f_θ. This adds more parameters to the optimization problem but has no consequences on the recognition model or the structure that can be imposed on the latent state.*

Remark 10. *The length and sampling times of the training trajectories need not be fixed. As long as they are known, we can interpolate the signals, compute the loss and perform optimization. However, if all trajectories have the same sampling times (or when simulating over the overarching set of sampling times from which all individual samples can be extracted), then all simulations can be run in parallel, which is computationally more efficient. Also, it would be beneficial to optimize all model parameters hierarchically: learn a first model on short training trajectories, which leads to a simpler optimization problem [151], then fine-tune it on increasingly long trajectories, e.g., by increasing n by a factor once an error threshold is reached. However, this and other procedures for increasing final accuracy necessitate extra implementation efforts and are not directly relevant for this work. In the following, we consider that the sampling times are the same for all trajectories and optimize for the full trajectory directly.*

5.4 Recognition models

In the case of partial observations, the initial condition x_0 in (5.2) is unknown and needs to be estimated jointly with f_θ. Estimating x_0 from partial observations is directly related to state estimation: while observers run forward in time to estimate the state asymptotically, we formulate this recognition problem as an optimization problem running backward in time to estimate the initial condition, akin to a receding horizon observer [152]. Therefore, the lens of observer design is well suited for investigating recognition models, though it has not been often considered. For example, whether the state can be estimated from the output is a precise notion in system theory called observability [42]:

Definition 3. *Initial conditions x_a, x_b are uniformly distinguishable in time $t_c > 0$ if for any input $u : \mathbb{R} \mapsto \mathbb{R}^{d_u}$*

$$y_{\mathrm{a,u}}(t) = y_{\mathrm{b,u}}(t) \; \forall \; t \in [0, t_c] \Rightarrow x_\mathrm{a} = x_\mathrm{b}, \tag{5.3}$$

where $y_{\mathrm{a,u}}$ (resp. $y_{\mathrm{b,u}}$) is the output of (5.1) given input u and initial condition x_a (resp. x_b). System (5.1) is observable (in t_c) if all initial conditions are uniformly distinguishable.

Hence, if (5.2) is observable, then $x(0)$ *is uniquely determined by y and u over $[0, t_c]$ for t_c large enough.* This assumption is necessary, otherwise there is no hope of approximating f with f_θ from the observations only. In the rest of the chapter, we indistinctly refer to a system as "observable" or "uniformly distinguishable".

In system identification, the unknown initial condition is usually directly optimized as a free variable: it needs to be optimized again for each new trajectory and cannot be used as such for prediction. Instead, we propose to estimate it from the

observations by learning a recognition model ψ_θ that links the output to the initial state, similarly to [152]. The term recognition model has been used to designate approximating the initial latent state conditioned on partial observations [65], [81], [82], [134]. For NODEs, recognition models have also been called augmentation strategies [33], [153]–[155]. Hence, we predict the initial latent state of the NODE as $x(0) = \psi_\theta(\bar{z}(t_c))$, where $\bar{z}(t_c)$ is the input of the recognition model. In what follows, we propose different methods for finding a suitable $\bar{z}(t_c)$. Concatenating the weights of f_θ and ψ_θ into θ leads to the modified problem:

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2d_y nN} \sum_{j=1}^{N} \sum_{i=0}^{n-1} \left| y^j(t_i) - \underline{y}^j(t_i) \right|^2 \\ s.t. \quad & \dot{x}^j = f_\theta(x^j, u^j) \qquad y^j = h(x^j, u^j) \\ & x^j(0) = \psi_\theta(\bar{z}^j(t_c)). \end{aligned} \tag{5.4}$$

5.4.1 General approaches

Some recognition methods have been proposed in the literature, not necessarily for system identification with NODEs, but rather for probabilistic [65] or generative [141] models. We draw inspiration from them and rewrite them to fit into our general framework, leading to the following.

Direct method

The most straightforward approach is to stack the observations and learn a mapping from

$$\bar{z}(t_c) = \underline{y}_{0:t_c} := (\underline{y}(0), \dots, \underline{y}(t_c)) \tag{5.5}$$

to the initial latent state. For nonautonomous systems, the first inputs should also be taken into account, which yields $\bar{z}(t_c) = (\underline{y}_{0:t_c}, u_{0:t_c})$. We denote this as the direct method. Variants of this approach have been used for approximating the distribution over the initial state conditioned on $\underline{y}_{0:t_c}$, e.g., for joint inference and learning of Gaussian process state-space models [65], [81], [82]. Augmentation strategies for NODEs [33], [153]–[155] are often particular cases with $t_c = 0$. However, for many nonlinear systems, this is too little information to estimate $x(0)$. There are few works on NODE-based system identification from partial observations, some of which train a recognition model from $\underline{y}_{-t_c:0}$ [141], [142], [156], or learn the dynamics of $\underline{y}_{0:t_c}$ [157].

As justified by the observability assumption, for t_c large enough this is all the information needed to estimate $x(0)$. However, for large t_c, the input dimension becomes arbitrarily high, which in turn aggravates the identification of ψ_θ. Also, the observations are not preprocessed in any way, though they may be noisy.

Recurrent recognition models

Latent NODEs [32], [134], [144] also use a recognition model to estimate the initial latent state from observations, though this may not be the state of an SSM. This recognition model is a Recurrent Neural Network (RNN) in [32] and an RNN combined with a second NODE in [134]. These methods go through the information contained in $(\underline{y}_{0:t_c}, u_{0:t_c})$ in backward time, process it through the current network parameters, then feed it to the recognition network ψ_θ, which is trained jointly with

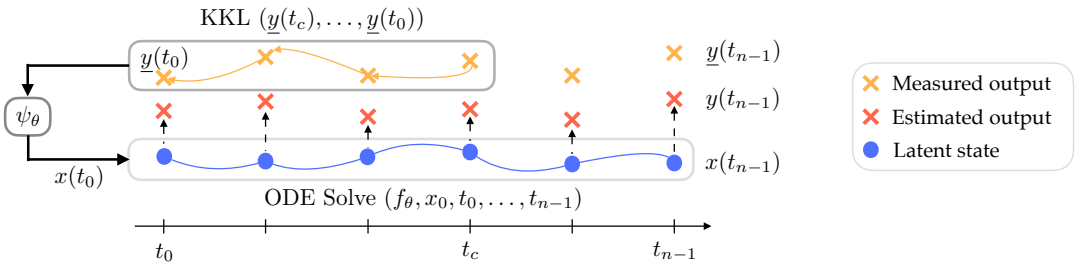


Figure 5.2: Illustration of the proposed method. The KKL observer runs backward in time over the observations on $[t_c, 0]$, estimates the initial latent state, then the NODE model runs forward to predict the following trajectory.

the (ODE-)RNN. We consider this baseline in the numerical results: we combine a Gated Recurrent Unit (GRU) of internal dimension d_z, run in backward time so that $\bar{z}(t_c)$ is the output of the GRU, and an output network ψ_θ. See Sec. 3.2.2 for more details on RNNs and GRUs in particular. We denote this method from [32] as RNN+. More complex models are now being developed on top of latent NODEs, e.g., [144], to improve the robustness of this combination of RNN recognition with ODE latent dynamics modeling w.r.t. irregular sampling times and gaps in the observations.

We now propose a novel type of recognition model based on nonlinear observer design, leading to different choices of $\bar{z}(t_c)$. See Table 5.1 for a summary of the proposed recognition methods.

5.4.2 KKL-based recognition models

As discussed in Sec. 3.4, the Kazantzis-Kravaris/Luenberger (KKL) observer [46], [47] is a particular algorithm for nonlinear state estimation. Roughly speaking, KKL observers rely on building a linear filter of the measurement: an auxiliary dynamical system of internal state z with known dimension d_z is simulated, taking the measurement as input and filtering it to extract the information it contains. The observer state satisfies

$$\begin{aligned} \dot{z} &= Dz + F\underline{y} \\ z(0) &= z_0 \end{aligned} \tag{5.6}$$

where $z \in \mathbb{R}^{d_z}$ with $d_z = d_y(d_x + 1)$ and z_0 is an arbitrary initial condition. In this system, $\underline{y}$ is the continuous-time measurement from (5.1), or an interpolation between the discrete observations $\underline{y}(t_i)$. The parameters D and F are chosen such that D is Hurwitz, i.e., all eigenvalues are in the left half-plane, and (D, F) is a controllable pair, i.e., the matrix $\begin{pmatrix} F & DF & \dots & D^{d_z-1}F \end{pmatrix}$ has full rank [158]. Thanks to the stability of (5.6), the internal state z "forgets" its arbitrary initial condition z_0 and converges asymptotically to a value depending on the history of $\underline{y}$. The speed of this convergence depends among others on the gain matrix D. Under certain conditions, this value uniquely determines, in turn, the value of the unmeasured state. More details are given in Sec. 3.4.

Autonomous systems

For autonomous systems, i.e., $u = 0$, it is shown in [47] that if the eigenvalues of D have sufficiently large negative real parts, then there exists an injective transformation

$\mathcal{T}$ and its left inverse $\mathcal{T}^*$ such that

$$|\underline{x}(t) - \mathcal{T}^*(z(t))| \underset{t\to\infty}{\longrightarrow} 0, \tag{5.7}$$

where $\underline{x}$ is the "true" underlying state of the system in the given state-space representation, meaning that $\mathcal{T}^*(z(t))$ with $z(t)$ from (5.6) is an observer for $\underline{x}(t)$. However, $\mathcal{T}^*$ cannot be computed analytically in general. Therefore, it has been proposed to learn it from full-state simulations [116], as further discussed in Ch. 7.

As D is Hurwitz and $\mathcal{T}$ resp. $\mathcal{T}^*$ transform x into z resp. z into x, running the observer (5.6) forward in time yields $\underline{x}(t_c) \approx \mathcal{T}^*(z(t_c)))$ for t_c large enough. However, this approach "throws away" the data $y_{0:t_c}$, which is used to simulate z on $[0, t_c]$ but not for dynamics estimation afterward, since the trajectory starts at $x(t_c) = \psi_\theta(z(t_c))$ with x the estimated NODE state. Therefore, we assume the system (5.1) is not backward but forward distinguishable, i.e., each state is uniquely determined by the previous values of the output and input; these two notions are equivalent when f is C^1 in particular. Then, as discussed in Sec. 3.4, the KKL observer also converges in backward time. Hence, running it backward in time on $[t_c, 0]$ yields $\underline{x}(0) \approx \mathcal{T}^*(z(0)))$ for t_c large enough; we then set $x(0) = \psi_\theta(z(0))$ and use all data $y_{0:t_n}$ for learning the NODE. Note that the transformations $\mathcal{T}$, $\mathcal{T}^*$ are not necessarily the same as the transformations for the forward observer, as the forward system may be more or less observable compared to the backward system.

Hence, we propose to train a recognition model $x(0) = \psi_\theta(z(0))$, where $z(0)$ is the result of (5.6) run backward in time for t_c from an arbitrary initial condition $z(t_c)$. This is further denoted as the KKL method, illustrated in Fig. 5.2. See Table 5.1 for a summary of the proposed methods.

Remark 11. *Directly learning an output predictor from partial observations with the KKL methodology has also been proposed in the literature, in particular under the name DeepKKL [159].Indeed, if the state-space representation of interest with state x is observable, then a KKL observer can be designed, that forgets its arbitrary initial condition z_0 and converges to $z(t)$ such that $z(t) \simeq \mathcal{T}(x(t))$ and $x(t) \simeq \mathcal{T}^*(z(t))$ for t large enough. If the function $\bar{h} = h \circ \mathcal{T}^*$ is approximated from trajectories of partial observations using standard supervised learning techniques, and if the true initial condition $z(0) = \mathcal{T}(x(0))$ is known, then simulating*

$$\dot{z} = Dz + F\bar{h}(z) \tag{5.8}$$

forward in time predicts the output y of the system. In practice the true initial condition is unknown, but a few samples of the measurement $y_{0:t_c}$ can be used to initialize the predictor: if t_c is large enough, thanks to the convergence of the observer, running the KKL observer (5.6) over $[0, t_c]$ yields a value $z(0)$ which is close to the true $\mathcal{T}(x(0))$. The rest of the trajectory $y(t)$, $t \geq t_c$ can then be predicted in open loop. This is an efficient way to initialize the system in z and builds an accurate output predictor, as investigated in DeepKKL [159].

If no prior knowledge about the underlying system is available, or no physical interpretation of the states is required, then learning such an output predictor might be a sufficient solution. The architecture of this predictor can be based on KKL observer design [159], on other theoretical results such as the existence of the Koopman operator [145], or on a general deep learning architecture justified by observability assumptions [160]. However, if the user is interested in a physically meaningful SSM, then a recognition model that maps the KKL state z to the physical coordinates is needed.

Nonautonomous systems

When extending the previous results to nonautonomous systems, $\mathcal{T}$ not only depends on $z(t)$ but also on time, in particular on the past values of u, and becomes injective for $t \geq t_c$ with t_c from the backward distinguishability assumption [114]. In the context of recognition models, this dependency on u over $[0, t]$ can be made explicit by running the observer (5.6) backward in time then training a recognition model $x(0) = \psi_\theta(\bar{z}(t_c))$ with $\bar{z}(t_c) = (z(0), u_{0:t_c})$. This is still denoted as the KKL method, for nonautonomous systems.

If the signal u can be represented as the output of an auxiliary system of inner state ω with dimension d_ω, then one can go back to a static transformation $\mathcal{T}$ with $z \in \mathbb{R}^{d_z}$ and $d_z = (d_y + d_u)(d_x + d_\omega + 1)$ by considering the extended dynamical system $X = (x, \omega)^\top$ with extended output $Y = (y, u)^\top$. We consider $q(X) = x$ as the observable part of this extended system and build a KKL observer for this part only. It is shown in [115] that the static observer

$$\begin{aligned} \dot{z} &= Dz + FY \\ z(0) &= z_0 \end{aligned} \tag{5.9}$$

leads to the same results as for autonomous systems. The time dependency in $\mathcal{T}$ disappears at the cost of a higher dimension: $d_z = (d_y + d_u)(d_x + d_\omega + 1)$. More details on KKL observers for nonautonomous systems and in particular functional KKL observers are provided in Sec. 3.4.2. This functional approach leads to an alternative recognition model denoted KKLu: $x(0) = \psi_\theta(z(0))$ where z is the solution of (5.9) simulated backward in time for t_c from an arbitrary initial condition, and d_ω is chosen large enough to generate u (e.g., $d_\omega = 3$ for a sinusoidal u).

Jointly optimizing the gain matrix

With any KKL-based recognition model, the choice of D in (5.6) – resp. (5.9) – is critical since it controls the convergence rate of z. Hence, we propose to optimize D jointly with θ by including the ODE on z in the optimization problem, as in [159]. This leads to a computational overhead but improves the performance.

In our experiments, we optimize D jointly with all other parameters once, then reuse the obtained value of D for all corresponding experiments. We set $F = \mathbb{1}_{d_z \times d_y}$ and initialize D with the following method. We compute the poles p_i of a Butterworth filter of order d_z and cut-off frequency $2\pi\omega_c$ and set each block of D as

$$D_i = \begin{cases} p_i & \text{if } p_i \text{ is real} \\ \begin{pmatrix} \Re p_i & \Im p_i \\ -\Im p_i & \Re p_i \end{pmatrix} & \text{otherwise} \end{cases} \tag{5.10}$$

such that D is a block-diagonal matrix, and its eigenvalues are the poles of the filter. This choice ensures that the pair (D, F) is controllable and that D is Hurwitz and has physically meaningful eigenvalues. Other possibilities exist, such as choosing D in companion form, as a negative diagonal, etc. However, we found that this strategy leads to the best performance for the considered use cases. We pick $\omega_c = 1$ for the systems of the recognition model benchmark and the harmonic oscillator with unknown frequency. For the robotics dataset, we initialize as $D = \text{diag}(-1, \ldots, -d_z)$. However, this choice is somewhat arbitrary, and the previous method with $\omega_c = 10$ shows similar performance. Principled methods for setting (D, F) are still needed for

easing the practical use of KKL observers; setting D to a HiPPO matrix [30], [161], [162] could be an interesting first step.

Conclusion on KKL recognition

For t_c large enough, the transformation $\mathcal{T}^*$ approximated by ψ_θ is guaranteed to exist for a known dimension d_z. The KKL observer also filters the information and provides a low-dimensional input to ψ_θ, which is expected to be easier to train. The RNN-based recognition models are close to learning a discrete-time observer with unknown dynamics: this is similar, but provides no theoretical argument for choosing the internal dimension of the RNN, no guarantee for the existence of a recognition model in form of an RNN, no physical interpretation for the behavior of the obtained observer, and leads to many more free parameters.

5.5 Enforcing physical knowledge

The aim of learning dynamical systems is generally to monitor, predict or control their behavior, tasks for which physically meaningful states are often necessary. One of the main advantages of the NODE formulation is that it is general: it can conceptually be applied to any observable system, contrarily to many classical, system-specific identification methods. However, it is also very straightforward to include physical knowledge in the optimization problem (5.4), which eases the generalization but also the interpretation of the obtained models. The extent of this physical knowledge varies depending on the use case, forming a broad spectrum illustrated in Fig. 5.1. As we shall show, NODEs can easily be adapted to move from one end of the spectrum to the other by changing (5.4): no knowledge at all will lead to one of infinitely many possible state-space representations, while for very strong priors, only one representation may remain feasible. Specific forms of prior knowledge have been proposed in the literature and are categorized below. Our contribution is to formulate a unifying view that can cover the whole spectrum of system identification illustrated in Fig. 5.1, on which these works can be positioned. Further, we combine this with partial observations, which have rarely been considered in the literature.

5.5.1 Regularizing priors

The first approach to include physical knowledge in NODEs is to add "regularizing" priors to the optimization problem. The most common case is when a prior model of the system, denoted f_0, is available from first principles. We can then learn a residual NODE on top of f_0, while penalizing the norm of f_θ to correct the prior only as much as necessary to fit the observations. This is investigated in [138], [139] for full state observations. It yields the modified optimization problem:

$$\begin{aligned} \min_{\theta} \quad & L(\theta) + \frac{\lambda}{2d_y nN} \sum_{j=1}^{N} \sum_{i=0}^{n-1} \left| f_\theta(x^j(t_i), u^j(t_i)) \right| \\ s.t. \quad & \dot{x}^j = f_0(x^j, u^j) + f_\theta(x^j, u^j) \qquad y^j = h(x^j, u^j) \\ & x^j(0) = \psi_\theta(\bar{z}^j(t_c)) \end{aligned} \tag{5.11}$$

where λ is a scaling factor set by the user. This scaling factor can be set by trial and error, then iterated over as in [138]; it characterizes how much the prior model should be trusted. Note that in the case of partial observations, this penalizes the norm of

the residuals along the trajectory given by the current dynamics, which could be somewhat of a circular dependency and lead to undesirable local optima; hence, we start with low values of λ.

Other quantities can be known a priori and enforced similarly to (5.11), such as the total energy of the system [163] or stability through a Lyapunov-inspired cost function [157], [164].

5.5.2 Structural constraints

Structural knowledge about the system can also be available and enforced by "structural" constraints, or a specific form of the optimization problem. As an example, if the system is known to be Hamiltonian, we can write:

$$\begin{aligned} \min_{\theta} \quad & L(\theta) \\ s.t. \quad & \dot{q}^j = \frac{\partial H_\theta}{\partial p}(x^j, u^j) \qquad \dot{p}^j = -\frac{\partial H_\theta}{\partial q}(x^j, u^j) \\ & x^j(0) = \psi_\theta(\bar{z}^j(t_c)) \qquad y^j = h(x^j, u^j) \end{aligned} \tag{5.12}$$

where $x = (q, p)$ are the generalized coordinates and H_θ is the approximate Hamiltonian. This yields a more difficult optimization problem, but improves the performance and interpretability of the model, particularly for long-term predictions. This line of work originates from [97]–[99] (see [100] for an overview) and has been extended to NODEs for Hamiltonian and port-Hamiltonian systems [165]–[167], but also to enforce more general energy-based structures [164], [168], the rules of electromagnetism [169] or those of thermodynamics [104]. However, these studies consider full state or even acceleration measurements to directly approximate the function of interest.

The above describes a family of optimization problems for dynamics learning: without prior knowledge (5.4), with "regularizing" priors (here a residual model) (5.11), and with "structural" priors (here a Hamiltonian structure) (5.12). While individual instances have been proposed in prior work, we regard the NODE formulation as a flexible framework for system identification: the whole spectrum of prior knowledge in Fig. 5.1 can be covered by mixing and matching such instances. We combine this view with recognition models for dealing with partial observations, and apply this empirically.

5.6 Experiments

We demonstrate the ability of the proposed approach to learn dynamics from partial observations with varying degrees of prior knowledge, illustrated in Fig. 5.1. Recognition models are critical when such prior knowledge is available: one is not looking for any latent space, but for one that has a specific structure, and learning a mapping from the observations to this specific latent state is difficult. Hence, we combine recognition with physical priors in the following. We first compare the different recognition models in combination with NODEs. We then provide an extensive case study on the harmonic oscillator, often used in the literature, learning its dynamics from partial measurements with increasing priors. Finally, we apply our approach to a complex use case with experimental data obtained on a robotic exoskeleton[1]. All models are evaluated w.r.t. their prediction capabilities: given $(y_{0:t_c}, u_{0:t_c})$ for a

[1]Code to reproduce the experiments is available at github.com/monabf/obsGP_recogNODE.git.

Method	$\bar{z}(t_c)$ for autonomous	$\bar{z}(t_c)$ for nonautonomous
$t_c = 0$	$\underline{y}(0)$	$(\underline{y}(0), u(0))$
direct	$\underline{y}_{0:t_c}$	$(\underline{y}_{0:t_c}, u_{0:t_c})$
RNN+	GRU over $\underline{y}_{t_c:0}$	GRU over $(\underline{y}_{t_c:0}, u_{t_c:0})$
KKL	KKL over $\underline{y}_{t_c:0}$	KKL over $\underline{y}_{t_c:0}$, concatenated with $u_{0:t_c}$
KKLu	n/a	functional KKL over $(\underline{y}_{t_c:0}, u_{t_c:0})$

Table 5.1: Summary of the proposed recognition methods for autonomous (center) and nonautonomous systems (right), all run backward in time. The recognition model ψ_θ is trained with $x(0) = \psi_\theta(\bar{z}(t_c))$. See Sec. 5.4 for more details.

number of test trajectories, we estimate the initial state, predict the further output, then compute the RMSE over the whole predicted trajectory.

Remark 12. *Note that there is a large part of randomness in the different experiments: the training and validation trajectories, the test set, and the noise are all realizations of random processes. Hence, results may vary, and obtaining consistent results to rigorously compare the different methods without any statistical variations would require running more experiments per setting.*

5.6.1 Pointers for training NODEs in practice

We start with a few practical recommendations to train NODEs for dynamical systems, which we have gathered during our experiments.

- If some of the initial conditions in the dataset are known, then imposing them reduces the learning effort by several orders of magnitude and should be used to build a first model which can learn at least the behavior from these known initial conditions.
- Preprocessing the data is an important first step. For example, consider filtering the available training trajectories if they are noisy, subsampling or cutting them. Experiment design is also crucial, as the training data should be rich enough to display all relevant types of system behavior.
- As shown in [151], it is preferable to learn from many short training trajectories rather than from fewer longer ones, as this smooths out the cost function and its gradient and makes the optimization problem more amenable. It is also preferable to avoid chaotic behaviors or regimes in which small changes in θ lead to large changes in the observed trajectory, which is more often the case with very long trajectories. For example, with the Hamiltonian model of the harmonic oscillator, the model converges for any sampling time and any number of training trajectories of 3 s, whereas it does not if they are 6 s long. Hence, rather use many short trajectories, if necessary by making small sub-trajectories from longer ones and adding continuity constraints as in [151].
- Normalization is a necessary step for most machine learning algorithms. To avoid numerical issues, we scale and unscale each input and output of the NODE so that each point that goes into the NN is normalized. However, scaling the inputs $x(t)$ is not straightforward when only partial observations $y(t)$ are

available. We choose to normalize the $y(\cdot)$ signal, then normalize the $x(\cdot)$ signal with the same scaler as $y(\cdot)$ on the measured dimensions and the mean of that scaler on the other dimensions, and to use these normalized quantities in all cost functions during optimization.

- Adding a control input to the NODE model has not been considered much in the literature, but it is crucial in control systems. When simulating the NODE, we interpolate linearly between the control and measurement samples. On the one hand, it is beneficial to have seen many values of the control input for learning a generalizable model. On the other hand, adding a control input increases the computational overhead by increasing the dimensionality of the NODE and by making simulations more time-consuming. Especially when inputs vary very fast, e.g., for random control inputs, the solver needs to take small steps to simulate the NODE, which is done at every optimization step, leading to an overall slower solution. Hence, the inputs and the training trajectories should be smooth enough to allow for the optimization to converge reliably.
- It can help to warm start the optimization, for example by learning the residuals of a prior model, or starting from the closest linear time-invariant (LTI) model.
- Ideally, the optimization should be run hierarchically: first training on short trajectories, then fine-tuning on longer ones.
- As often in deep learning, some hyperparameters such as the learning rate scheduler, weight decay, optimization routine, or minibatch size influence the performance and should be chosen carefully.
- Finally, the NN design chosen to approximate the dynamics should be flexible enough to represent the chosen system, but needs not be as large as for some other applications in machine learning. The problem still needs to be over-parametrized enough such that there are many equally good local optima that we can hope to find; however, after a while adding parameters only enables the optimization process to reach the same plateau in fewer iterations, but each iteration takes longer, so it is not advantageous anymore. We recommend `Tanh` or `SiLU` as activation functions, which are flexible but smooth enough to represent dynamical systems. Consider using a parsimonious NN design or a structure that is adapted to the setting at hand, for example a Convolutional Neural Network (CNN) if spatial dependencies should be captured.

This list is not exhaustive, but can help a user apply the proposed framework to experimental data.

5.6.2 Benchmark of recognition models

In this section, we illustrate that the proposed recognition models can estimate the initial condition of a system from partial and noisy observations. We simulate the three following systems, the underlying physical state serves as ground truth.

Earthquake model

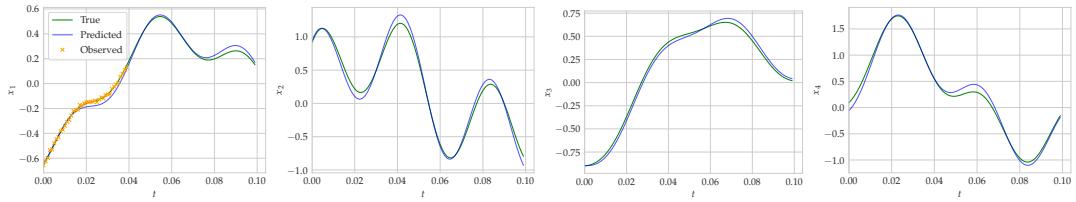
A simplified model of the effects of an earthquake on a two-story building is presented in [170], and an NODE is trained for it in [171]. This model reads

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{k}{m}(x_3 - 2x_1) - F_0\omega^2\cos(\omega t) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{k}{m}(x_1 - x_3) - F_0\omega^2\cos(\omega t) \\
y &= x_1 + \epsilon,
\end{aligned} \tag{5.13}$$

where x_1 and x_3 are the positions of the first resp. second floor, x_2 resp. x_4 their velocities, $F_0\omega^2\cos(\omega t)$ is the perturbation caused by the earthquake and only x_1 is observed, corrupted by Gaussian noise of variance $\sigma_\epsilon^2 = 10^{-4}$. We consider the oscillation caused by the earthquake as a disturbance, which is known when simulating training trajectories and unknown to the recognition model: we estimate $x(0)$ from $\underline{y}_{0:t_c}$ only.

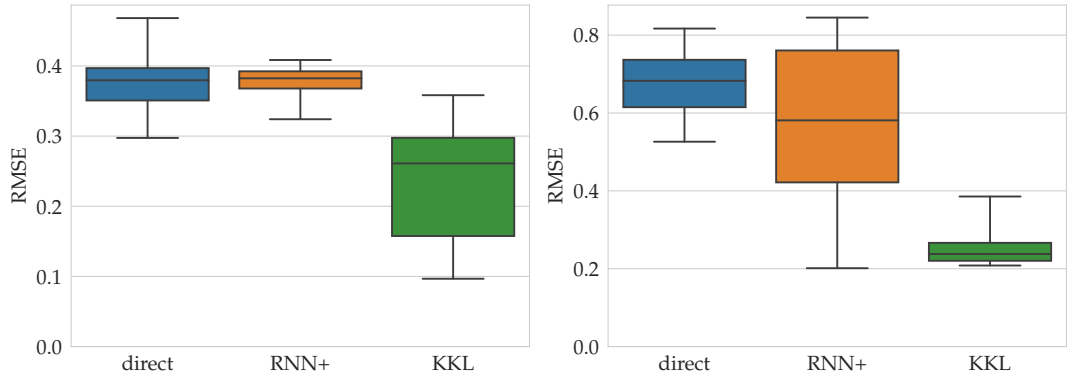
We aim to learn a recognition model that estimates $x(0)$ using only $\underline{y}_{0:t_c}$ with the methods described above. We set $t_c = 40 \times \Delta t = 40 \times 0.03 = 1.2\,\mathrm{s}$, which seems enough to reconstitute the initial condition (after trial and error), $N = 50$ (each sample corresponds to a random initial condition, random F_0 and random ω), $n = 100$. We design ψ_θ (and possibly f_θ) as a fully connected feed-forward network, i.e., a multilayer perceptron, with two hidden layers containing 50 neurons each, and two fully connected input and output layers with bias terms. The RNN+ model is set to have the same internal dimension $d_z = d_y(d_x + 1) = 5$ as the KKL model. We notice that t_c large enough and enough parameters in ψ_θ, i.e., enough flexibility of the model, are needed for good generalization performance. Also, we pick the sampling time $\Delta t = 0.03\,\mathrm{s}$ low enough such that the obtained trajectories are reasonably smooth, otherwise analyzing the results quantitatively becomes difficult due to interpolation errors becoming large.

We train ten recognition models with each proposed method. For evaluation, we randomly select 100 test trajectories (also 3 s) with random initial conditions and random input oscillation, estimate the initial condition from $y_{0:t_c}$, then compute the RMSE over the predicted output. We try two settings: either learning a full NODE model, or having a known dynamics model in which only the main parameter k/m is optimized jointly with the recognition model. This corresponds to the far left (black-box ODE) and far right (parametric model) ends of the spectrum in Fig. 5.1. In our example, we have $k/m = 10$, but we initialize its estimate to a random value in the interval $[8, 12]$. As usual, this problem is not well-posed and there are many local optima. Therefore, we can only hope to converge to a good estimate by starting from a reasonable guess of the main parameter. All other aspects are fixed, including the optimization routine, though it has been shown that for parametric optimization, trust region optimization routines with multiple starts often lead to better results [20]. For the full NODE model, we evaluate the different recognition models by computing the RMSE on the predicted output only, since the coordinate system for $x(t)$ is not fixed and a different coordinate system is found in each experiment. For the parametric model, we compute the RMSE on the whole estimated trajectory over all test scenarios, since the coordinate system for $x(t)$ is fixed. The results are shown in Figure 5.4. We observe that the KKL-based models achieve higher performance, which seems to



(a) x_1 (b) x_2 (c) $\dot{x}_1$ (d) $\dot{x}_2$

Figure 5.3: Test trajectory of the parametric earthquake model with KKL recognition: the initial condition is estimated from $\underline{y}_{0:t_c}$ jointly with the model parameters.



(a) Full NODE: output error (b) Parametric model: full state error

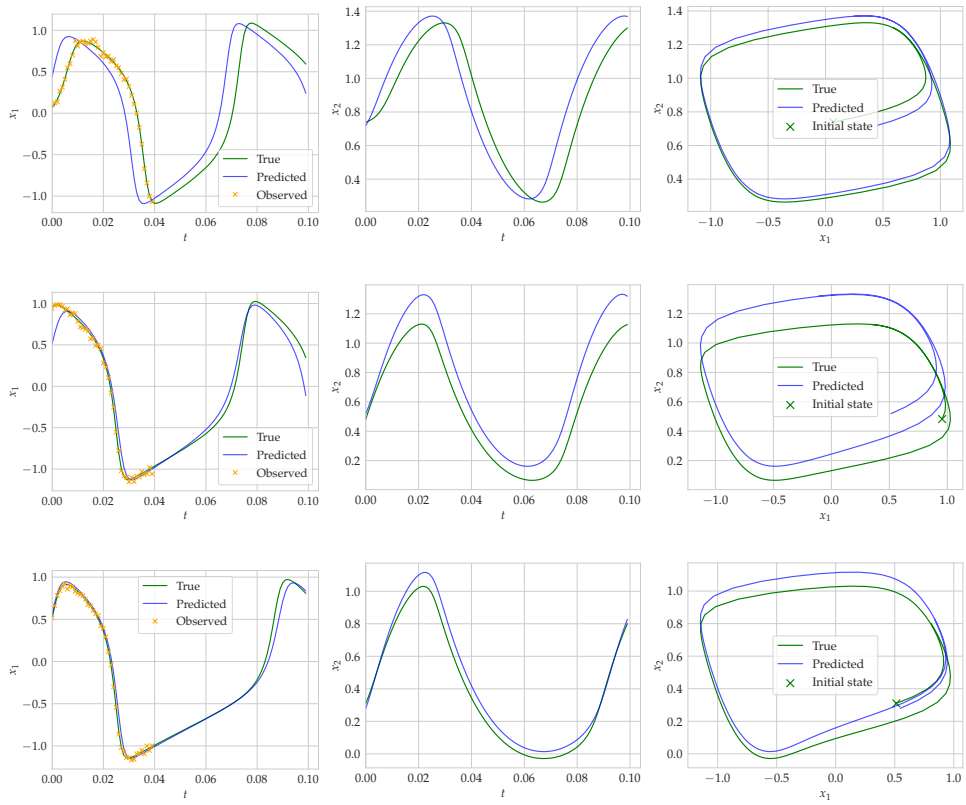
Figure 5.4: Results of the obtained earthquake recognition models. We show the RMSE on the prediction of the output when a full NODE model is learned (left column) and of the whole test trajectories when a parametric model is learned (right column). Ten recognition models were trained with the methods direct (left), RNN+ (center), and KKL (right). The direct method with $t_c = 0$ is not shown here for scaling, but the mean RMSE is over 0.6.

indicate that the optimization problem based on $z(0)$ is better conditioned than that based on $y_{0:t_c}$. An example test trajectory with KKL recognition is shown in Fig. 5.3 for the parametric model, as it is in the same coordinate system as the ground truth.

FitzHugh-Nagumo model

This model represents a relaxation oscillation in an excitable system. It is a simplified representation of the behavior of a spiking neuron: an external stimulus is received, leading to a short, nonlinear increase of membrane voltage then a slower, linear recovery channel mimicking the opening and closing of ion channels [172]. The dynamics are written as

$$\begin{aligned} \dot{v} &= \frac{1}{\epsilon}(v - v^3 - u) + I_{ext} \\ \dot{u} &= \gamma v - u + \beta \\ y &= v + \epsilon, \end{aligned} \tag{5.14}$$



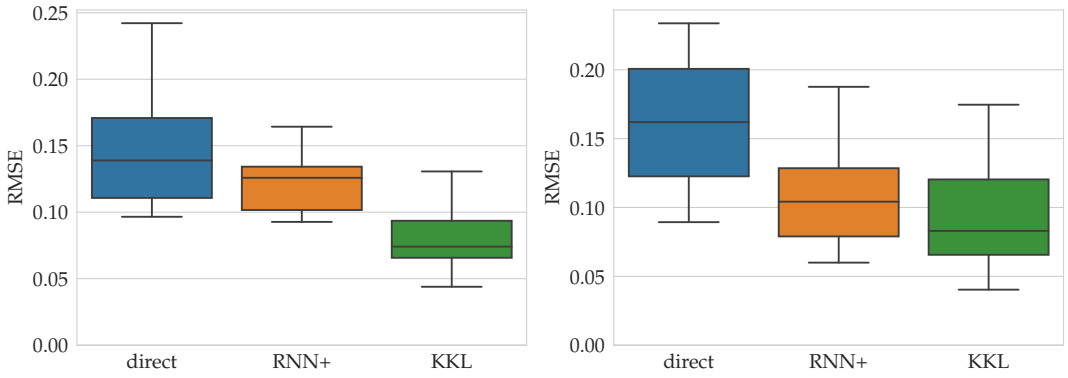
(a) u (b) v (c) Phase portrait

Figure 5.5: Test trajectory of the parametric FitzHugh-Nagumo model: the initial condition is estimated from $\underline{y}_{0:t_c}$ jointly with the model parameters. We use direct (top), RNN+ (center) and KKL (bottom) recognition, on three random but similar test trajectories.

where v is the membrane potential, u is the value of the recovery channel, I_{ext} is the value of the external stimulus (here a constant), $\epsilon = 0.1$ is a time scale parameter, and $\gamma = 1.5$, $\beta = 0.8$ are kinetic parameters. Only v is measured, corrupted by Gaussian measurement noise ϵ of variance $\sigma_\epsilon^2 = 5 \times 10^{-4}$.

We aim to learn a recognition model that estimates $(v(0), u(0))$ using $\underline{y}_{0:t_c}$ and I_{ext} with the methods described above. All parameters are set as for the earthquake model above.

As above, we train ten recognition models with each proposed method and evaluate each on 100 test trajectories of random initial conditions and random input oscillation. We either learn a full NODE model or a parametric model for which we estimate the main dynamic parameters ϵ, β, and γ jointly with the recognition model, initialized randomly in $[0.05, 0.15]$, $[0.75, 2.25]$ and $[0.4, 1.2]$ resp. . We evaluate as above. The results are shown in Fig. 5.6; we observe once again that the KKL-based methods lead to lower error. An example test trajectory with different recognition models is shown in Fig. 5.5 for the parametric model, as it is in the same coordinate system as the ground truth.



(a) Full NODE: output error

(b) Parametric model: full state error

Figure 5.6: Results of the obtained FitzHugh-Nagumo recognition models. We show the RMSE on the prediction of the output when a full NODE model is learned (left column) and of the whole test trajectories when a parametric model is learned (right column). Ten recognition models were trained with the methods direct (left), RNN+ (center), and KKL (right). The direct method with $t_c = 0$ is not shown here for scaling, but the mean RMSE is over 0.4.

Van der Pol oscillator

Consider the nonlinear Van der Pol oscillator of dynamics

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= \mu(1 - x_1^2)x_2 - x_1 + u \\ y &= x_1 + \epsilon, \end{aligned} \tag{5.15}$$

where x_1, x_2 are the states, $u = 1.2\sin(\omega t)$ is a sinusoidal control input, and $\mu = 1$ is a damping parameter. Only x_1 is measured, corrupted by Gaussian measurement noise ϵ of variance $\sigma_\epsilon^2 = 10^{-5}$.

We aim to learn a recognition model that estimates $x(0)$ using $\underline{y}_{0:t_c}$ and $u_{0:t_c}$ with the methods described above. All parameters are set as for the earthquake model above. Since u is a sinusoidal control input of varying frequency, it can be generated by

$$\begin{aligned} \dot{\omega}_1 &= \omega_2, \\ \dot{\omega}_2 &= -\omega_3\omega_1, \\ \dot{\omega}_3 &= 0 \end{aligned} \tag{5.16}$$

and $u = \omega_1$, where ω_1, ω_2 are the internal states of the sinusoidal system and $\omega_3 > 0$ is its frequency. Hence, we can use the KKLu recognition model with $d_\omega = 3$. The RNN+ model is set to have the same internal dimension d_z as the KKLu model.

As with the previous systems, we train ten recognition models with each proposed method and evaluate the results on 100 test trajectories with random initial conditions and random input frequency. We also consider either a full NODE model, or a parametric model for which μ is jointly estimated, from a random initial guess in $[0.5, 1.5]$. We evaluate as above. The corresponding box plots are shown in Fig. 5.9. We observe that in both settings, the performance with the different recognition models is very similar. Thus, the hierarchy of the different recognition models slightly

varies with other hyperparameters, such as the level of measurement noise, as seen in the ablation study (Fig. 5.11). An example test trajectory with different recognition models is shown in Fig. 5.7 for the parametric model, as it is in the same coordinate system as the ground truth. The different levels of noise are illustrated in Fig. 5.8.

Ablation studies

In the previous benchmark, we choose all hyperparameters such that the comparison is as fair as possible. For example, ψ_θ is the same neural network in all cases, the dimension of the internal recognition state is the same for the RNN+ and KKL baselines (d_z of the standard KKL for autonomous systems, d_z of the functional KKL for nonautonomous systems). We now investigate the impact of two of the main parameters on the performance of each approach for the full NODE models: t_c and σ_ϵ^2, the variance of the Gaussian measurement noise added on top of the true output to create the training data.

For the study on t_c, we focus on the earthquake system. We run the same experiments as before with t_c in $\{5, 10, 20, 40, 60, 100\} \times \Delta t$, where $\Delta t = 0.03\,\mathrm{s}$. As depicted in Fig. 5.10, when t_c is too low it becomes difficult to estimate $x(0)$ from the information contained in $\underline{y}_{0:t_c}$: the system loses observability. It seems the threshold of observability is around $t_c = 30\Delta t = 0.9\,\mathrm{s}$, since the RMSE over the test trajectories stabilizes for higher values. For this system, the KKL method reaches the lowest error and keeps improving for higher values of t_c: the higher t_c, the more the observer has converged, the closer the relationship $x(0) \approx \mathcal{T}^*(\bar{z}(0))$ is and the easier it seems to learn ψ_θ. For the other methods, t_c seems to have less influence once the threshold of observability is reached since there is no notion of convergence over time.

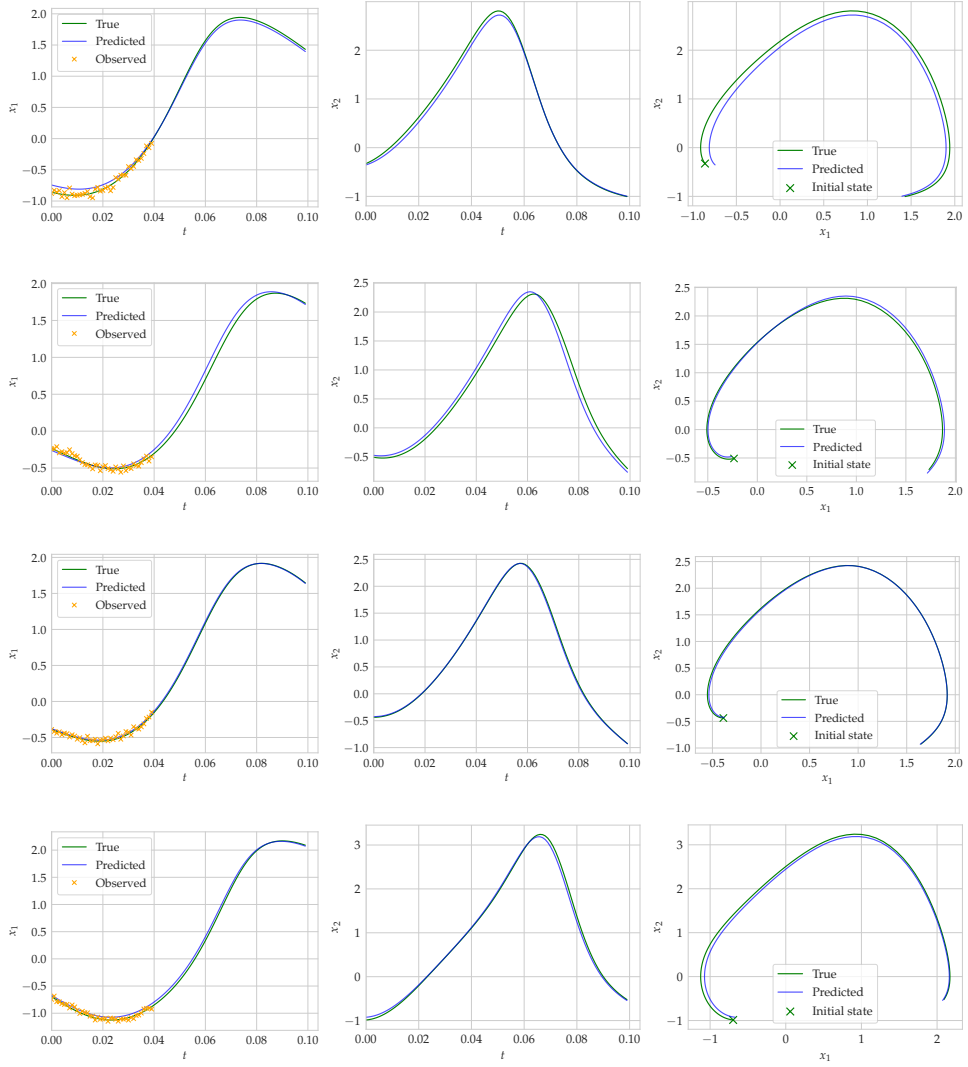
For the study on σ_ϵ^2, we focus on the Van der Pol oscillator. We test for values in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and obtain the results in Fig. 5.11. As expected, the higher the measurement noise variance, the higher the prediction error on the test trajectories. We again observe a threshold effect, under which further reduction of the noise variance leads to little improvement in the prediction accuracy. Note that for the KKL-based methods, we optimized D once for each noise level from the same initial value, then used this optimized value for all ten experiments. If D is only optimized for a specific noise level, then the performance is degraded at the others, for which this value might filter too much or too little.

5.6.3 Harmonic oscillator with increasing priors

We now illustrate how NODEs with KKL-based recognition can be combined with physics-aware approaches to cover the different degrees of structure illustrated in Fig. 5.1. We simulate an autonomous harmonic oscillator with unknown frequency:

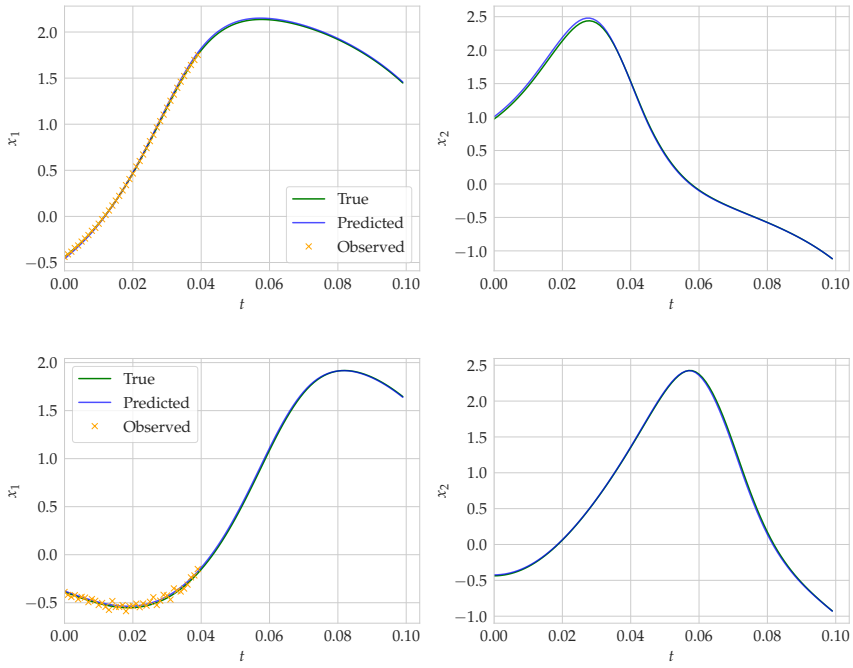
$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\omega^2 x_1, \end{aligned} \tag{5.17}$$

where $\omega^2 > 0$ is the unknown frequency of the oscillator and $y = x_1$ is measured, corrupted by Gaussian noise of standard deviation $\sigma = 0.01$. Various designs have been proposed to identify both the state and the model, i.e., the frequency, for example subspace methods, or extended state-space models combined with a nonlinear observer; see [137] and references therein. We demonstrate that our unifying framework can solve this problem while enforcing increasing physical knowledge. The results



(a) x_1 (b) x_2 (c) Phase portrait

Figure 5.7: Test trajectory of the parametric Van der Pol model: the initial condition is estimated from $\underline{y}_{0:t_c}$ jointly with the model parameters. We use direct (top), RNN+, KKL, and KKLu (bottom) recognition, on four random but similar test trajectories with $\sigma_\epsilon^2 = 10^{-3}$.



(a) x_1 (b) x_2

Figure 5.8: Test trajectory of the parametric Van der Pol model with KKL recognition. The Gaussian measurement noise has variance $\sigma_\epsilon^2 = 10^{-5}$ above, 10^{-3} below.

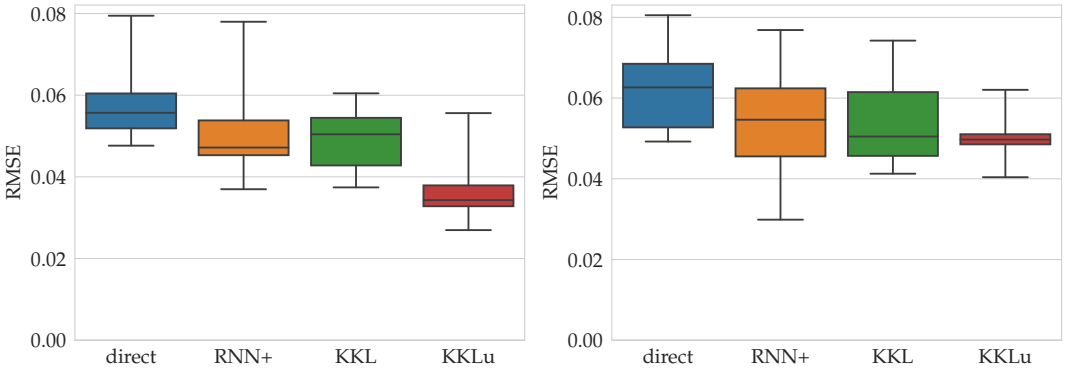
are depicted in Table. 5.2 with a KKL recognition model, $\omega = 1$, $N = 20$ trajectories of 3 s for training and hundred trajectories of 9 s for testing.

The obtained results are illustrated in Figure 5.12 with the KKL recognition model. We show the prediction of a random test trajectory with random initial state: the output $\underline{y}_{0:t_c}$ is measured for this trajectory, used by the recognition model to estimate $x(0)$. Then, the learned NODE is simulated to predict the whole state trajectory for 500 time steps, i.e., ten times longer than the training time to illustrate the long-term behavior. For the quantitative results presented in Table. 5.2, we predict on test trajectories of 150 time steps, i.e., three times the training time. These make the long-term performance difference due to the degree of prior knowledge less visible, but lead to more consistent and quantitatively comparable results (with the long test trajectories, the interquartile range of the experiments is very large due to error accumulation which can possibly diverge over a long prediction horizon).

We train the dynamics and recognition model ten times in each setting, for recognition models direct, RNN+, and KKL. The mean RMSE over hundred short test trajectories is given in Table 5.2.

No structure

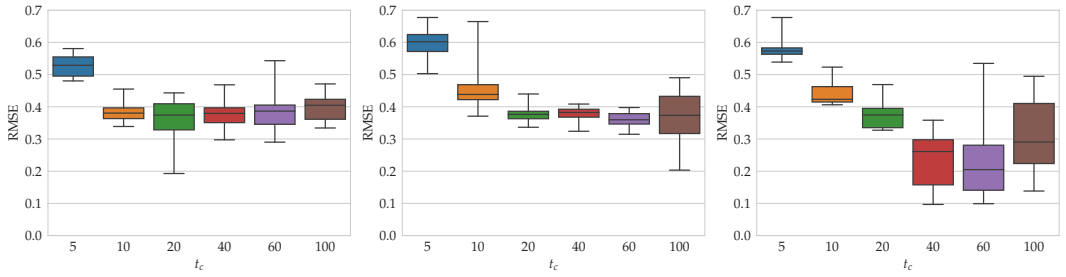
First, the NODE is trained without any structure (Fig. 5.12a), which leads to one of many possible state-space models: it fits the observations in x_1 but finds another coordinate system for the unmeasured state, as expected for general latent NODEs. It also does not conserve energy, which is not surprising when no structure is imposed, as discussed e.g., in [98].



(a) Full NODE: output error

(b) Parametric model: full state error

Figure 5.9: Results of the obtained Van der Pol recognition models. We show the RMSE on the prediction of the output when a full NODE model is learned (left column) and of the whole test trajectories when a parametric model is learned (right column). Ten recognition models were trained with the methods direct (left), RNN+, KKL, and KKLu (right). The direct method with $t_c = 0$ is not shown here for scaling, but the mean RMSE is over 0.6.



(a) direct

(b) RNN+

(c) KKL

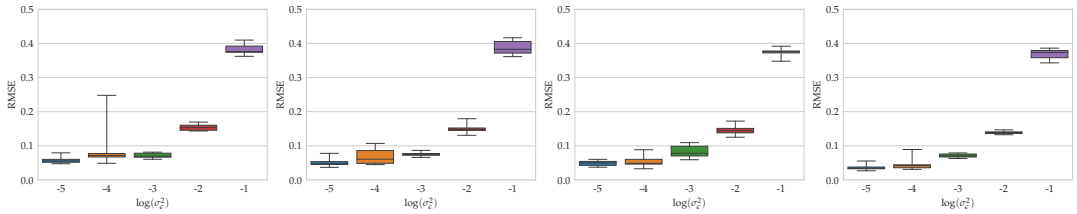
Figure 5.10: Training an NODE and a recognition model for the earthquake model for different lengths of t_c (in number of time steps). We train ten models for each method.

Hamiltonian state-space model

We now assume the user has some physical insight about the system at hand (Fig. 5.12b): it derives from a Hamiltonian function, i.e., there exists H such that

$$\dot{x}_1 = \frac{\partial H}{\partial x_2}(x), \qquad \dot{x}_2 = -\frac{\partial H}{\partial x_1}(x). \tag{5.18}$$

We approximate H directly with a neural network H_θ of weights θ, such that the NODE has form (5.18), as in (5.12) with $x_1 = q$, $x_2 = p$. This formulation enforces the constraint that the dynamics derive from a Hamiltonian function, whose choice is free. In that case, we do not necessarily find the "physical" state-space realization, as several Hamiltonian functions can fit the data. However, the obtained state-space model conserves energy due to the Hamiltonian structure: we learn the dynamics up to a symplectomorphism [34].



(a) direct (b) RNN+ (c) KKL (d) KKLu

Figure 5.11: Training an NODE and a recognition model for the Van der Pol model for different values of σ_ϵ^2. We train ten models for each method.

Imposing position and velocity

We now impose a somewhat stronger structure (Fig. 5.12c):

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\nabla H(x_1), \tag{5.19}$$

where only the dynamics of x_2 need to be learned. This enables the NODE to recover both the initial state and the unknown part of the dynamics in the imposed coordinates while also conserving energy, as this is a particular case of Hamiltonian dynamics with Hamiltonian function $\frac{1}{2}x_2^2 + H(x_1)$.

Parametric system identification

We now directly learn a parametric model of the harmonic oscillator (Fig. 5.12d):

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\omega^2 x_1,$$

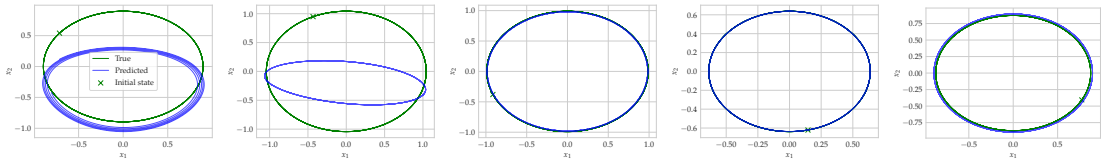
where $\omega > 0$ is the unknown frequency. We approximate ω with a parameter θ, which is initialized randomly in $[0.5, 2]$. We obtain excellent results with this method, as θ is estimated correctly up to 10^{-2}, and the trained recognition model gives satisfying results. This demonstrates that our framework can recover both the dynamics and the recognition model in the physical coordinates imposed by the parametric model from partial and noisy measurements in this simple use case.

Extended state-space model

Finally, we consider the extended state-space model (Fig. 5.12e)

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -x_3 x_2, \qquad \dot{x}_3 = 0 \tag{5.20}$$

where $x_3 = \omega^2$ is a constant state representing the unknown frequency. In this case, the dynamics are completely known and only the recognition model is left to train, to estimate the initial condition $x(0) \in \mathbb{R}^3$, where $x_3(0)$ is the unknown frequency. This is the same degree of structure as the parametric model: the dynamics are known up to the frequency. However, it also is a more open problem: since we learn a recognition model for $x(0) \in \mathbb{R}^3$, for each new trajectory, we estimate a new value of the frequency $x_3(0)$, which was considered the same across all trajectories for the previous methods. Therefore, with this setting, we also obtain models that can predict energy-preserving trajectories in the physical parameters, but with lower accuracy due to this extra degree of freedom.



(a) No structure (b) Hamiltonian (c) $\dot{x}_1 = x_2$ (d) Parametric (e) Extended

Figure 5.12: Structured NODEs for the harmonic oscillator, with KKL recognition and increasing priors. The true trajectory is in green, and the prediction of a long trajectory (30 s) in blue to illustrate the long-term accuracy. Increasing structure yields a more interpretable, but also a more accurate model, except for (e) which solves a more open problem (a new frequency is estimated for each trajectory).

Method	(a)	(b)	(c)	(d)	(e)
direct	0.040 (0.011)	0.050 (0.033)	**0.035** (0.008)	**0.029** (0.005)	0.080 (0.041)
RNN+	0.057 (0.014)	0.055 (0.012)	0.048 (0.003)	0.037 (0.006)	0.052 (0.011)
KKL	**0.036** (0.010)	**0.042** (0.011)	**0.036** (0.004)	0.032 (0.003)	**0.049** (0.003)

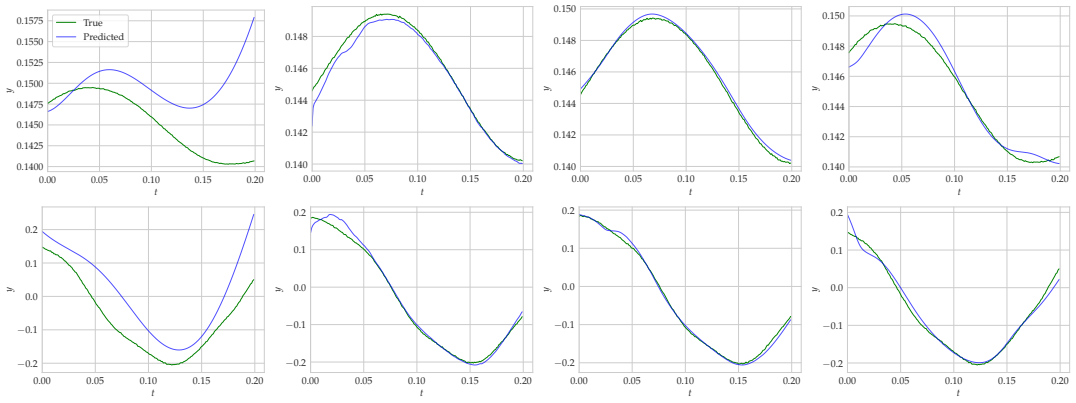
Table 5.2: Recognition models for the harmonic oscillator. We train ten models for each setting, then compute the median and interquartile range (in parentheses) of the RMSE on the predicted output for a hundred short test trajectories of 9 s. In most cases, KKL recognition leads to more accurate predictions.

The results in Fig. 5.12 illustrate that NODEs with recognition models can incorporate gradual priors for learning SSMs from partial and noisy observations. Note that standard methods tailored to the harmonic oscillator may perform better, however, they are not as general nor as flexible.

5.6.4 Experimental dataset from a robotic exoskeleton

We demonstrate the performance of the proposed framework on a real-world dataset.

Figure 5.13: Robotic exoskeleton by Wandercraft.



(a) Prior model (b) No structure (c) $\dot{x}_{1/3} = x_{2/4}$ (d) Residuals and (c)

Figure 5.14: Structured NODEs and KKL recognition on the robotics dataset. The dynamics and KKL recognition models are learned from 265 measurements of 0.2 s with inputs of varying frequencies. We test on 163 trajectories of 0.2 s from the same input frequencies to evaluate data fitting in the trained regime, and compute the RMSE: we obtain resp. 5.6 (a), 0.16 (b), 0.18 (c), 0.31 (d). We show one such test trajectory (x_1 top row, x_4 bottom row) from an unknown initial condition.

Recognition model	Short rollouts in trained regime	Long rollouts with unseen frequencies	Long rollouts with EKF: $y = (x_1, x_4)$	Long rollouts with EKF: $y = x_1$
direct	0.11	0.58	0.12	0.44
RNN+	0.15	0.54	0.15	0.31
KKL	0.17	0.60	0.12	0.37
KKLu	0.18	0.65	0.11	0.34

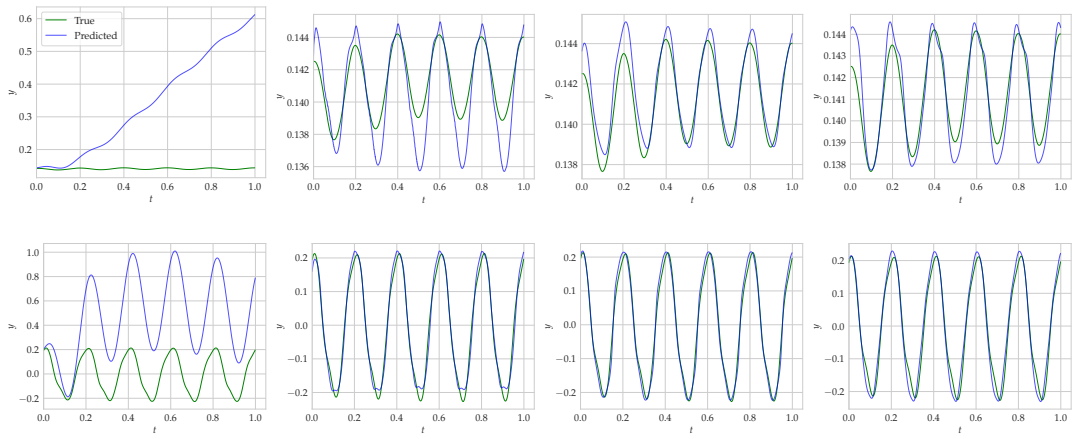
Table 5.3: RMSE on test trajectories for the robotics dataset while imposing $\dot{x}_{1/3} = x_{2/4}$. We trained only one model per method; hence, the results only illustrate that all methods are comparable.

Data collection

We use a set of measurements collected from a robotic exoskeleton at Wandercraft[2], presented in [173] and Fig. 5.13. This robot features mechanical deformations at weak points of the structure that are neither captured by computer-assisted design modeling nor measured by encoders. These deformations, when measured by a motion capture device, can be shown to account for significant errors in foot placement. Further, they exhibit nonlinear spring-like dynamics that complicate control design.

The dataset is obtained by fixing the robot basin to a wall and sending a sinusoidal excitation signal to the front hip motor at different frequencies between 2 Hz and 16 Hz. The sagittal hip angle is measured by an encoder, while the angular velocity of the thigh is measured by a gyroscope. In [173], first results are obtained using linear system identification: the observed deformation is modeled as a linear spring in the hip, and this model is linearized around an equilibrium point, then its parameters are

[2]More details on the robot, the dataset and the methods applied at Wandercraft are provided in Section 4.1.2.1 of [173].



(a) Prior model (b) No structure (c) $\dot{x}_{1/3} = x_{2/4}$ (d) Residuals and (c)

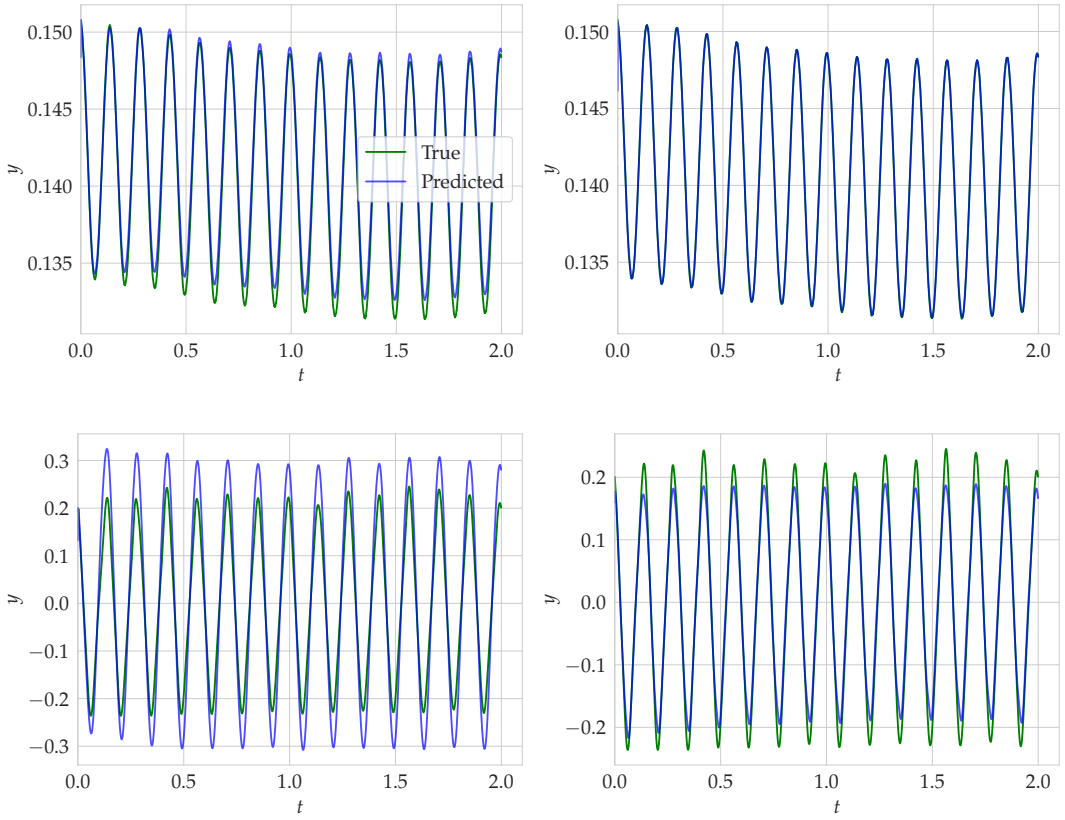
Figure 5.15: Structured NODEs and KKL recognition on the robotics dataset. After training the NODE on trajectories of 0.2 s from a subset of the input frequencies, we also test on 52 trajectories of 2 s from other input frequencies, to evaluate generalization capabilities (cut at 1 s on plots for visibility). Computing the prediction RMSE for the different structure settings yields: 110 (a), 0.72 (b), 0.66 (c), 1.2 (d). We show one such test trajectory (x_1 top row, x_4 bottom row) from an unknown initial condition.

identified. These estimates are sufficient for tuning a robust controller to compensate for the deformation. We aim to learn a more accurate model of this dynamical system of dimension $d_x = 4$, where $y = (x_1, x_4)$ is measured, by identifying the nonlinear deformation terms.

Data processing

We start by preprocessing the signal: for each input frequency and corresponding trajectory, we compute the FFT of y, apply a Gaussian window at $f_c = 50$ Hz on the spectrum, then apply an inverse FFT and slice out the beginning and the end (100 time steps) of each signal to get rid of the border effects. For u, which is not very noisy, we rather apply a Butterworth filter of order 2 and cut-off frequency 200 Hz. We cut the long trajectories for each input frequency in slices of 200 samples, and stack these training trajectories of length 0.2 s together to form our set of training trajectories. Hence, all trajectories have the same sampling times and can be simulated in parallel easily. We choose a length of 0.2 s because it seems long enough to capture some of the dynamics even in the low-frequency regime, but also short enough to remain acceptable in the high-frequency regime, as discussed in Remark 10.

Normalization is also an important aspect of the implementation: all losses and evaluation metrics are scaled to the same range, so that all loss terms play a similar role and remain within a similar range. This ensures that the values on which the optimization is based are always numerically tractable for the chosen solver. Different scaling possibilities are discussed in Sec. 5.2.3 of [18]. In our case, since we do not know in advance the values that $x(t)$ will take, we compute the mean and standard deviation of the samples in $y(t)$ and $u(t)$ and scale all outputs $y(t)$ resp. inputs $u(t)$ according to these. We also scale all states $x(t)$ or derivatives $\dot{x}(t)$ using the scaler on $y(t)$ for the dimensions that are measured (x_1 and x_4), and a mean of the scaler on



(a) Prior model

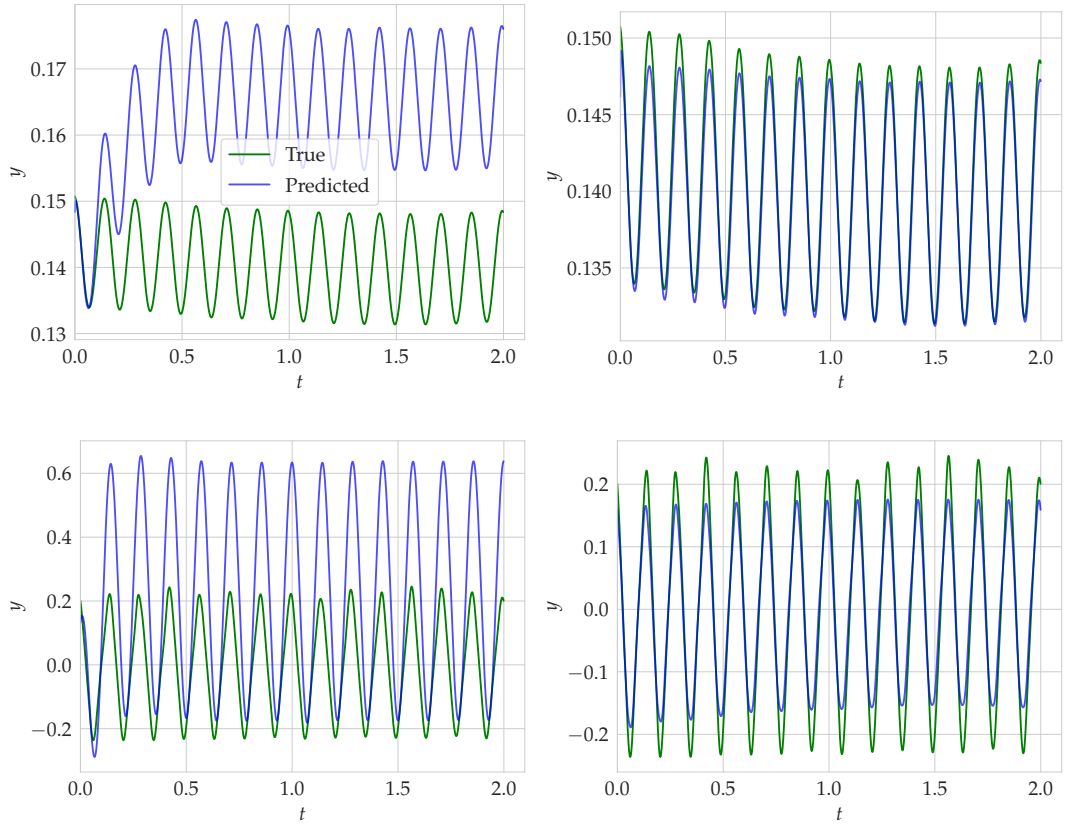
(b) NODE with $\dot{x}_{1/3} = x_{2/4}$

Figure 5.16: State estimation with an EKF on the robotics dataset, with $y = (x_1, x_4)$. After training the NODE with KKL recognition while imposing $\dot{x}_{1/3} = x_{2/4}$, we run the EKF on long test trajectories from unseen input frequencies. Both prior and learned models are able to reconstitute the output (x_1 top row, x_4 bottom row). However, the performance difference indicates that the states estimated by the NODE are indeed more accurate and can be used for downstream tasks.

$y(t)$ for the other dimensions. This arbitrary choice seems to ensure that all scaled values of $x(t)$ stay within a reasonable range.

Evaluation

We investigate three settings: no structure (b), imposing $\dot{x}_1 = x_2$ and $\dot{x}_3 = x_4$ (c), and learning the residual of the prior linear model on top of this structure as in [138], by using $f = f_{lin} + f_\theta$ as the dynamics, where f_{lin} is the linear prior (d). We train from a subset of input frequencies: $\{2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 11, 13, 15\}$ Hz. We use a random subset of $N = 265$ training trajectories, and a subset of validation trajectories for early stopping. We then evaluate on 163 test trajectories of 0.2 s from these input frequencies, to evaluate data fitting in the trained regime. We also evaluate on 52 longer (2 s) test trajectories from other input frequencies, to evaluate the interpolation capabilities of the learned model: $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17\}$ Hz. We use the Adam [90] optimizer with a decaying learning rate starting at 8×10^{-3} for the first two settings, 5×10^{-3} for the third setting. For all recognition models, we set $t_c = 0.1$ s. For KKL recognition, we set $d_z = 10$, $F = \mathbb{1}_{d_z \times d_y}$, which yields 110 for the dimension of $(z(t_c), u_{0:t_c})$, and optimize D for each setting after initialization at



(a) Prior model

(b) NODE with $\dot{x}_{1/3} = x_{2/4}$

Figure 5.17: State estimation with an EKF on the robotics dataset, with $y = x_1$. After training the NODE with KKL recognition while imposing $\dot{x}_{1/3} = x_{2/4}$, we run the EKF on long test trajectories from unseen input frequencies. When measuring only x_1, the EKF using the prior model is off (x_1 top row, x_4 bottom row), while it provides reasonable estimates in most frequency regimes when using the learned model.

diag$(-1, \dots, -10)$. For KKLu recognition, we set $d_z = 50$, $F = \mathbb{1}_{d_z \times d_y}$ and optimize D after initialization at diag$(-2, \dots, -100)$. For direct and RNN+, we use the same information contained in $(\underline{y}_{0:t_c}, u_{0:t_c})$ and the same size of the RNN+ latent state as for KKLu, i.e., dimension 50. Both recognition and dynamics models are feed-forward networks with five hidden layers of resp. 50 and 100 neurons with SiLU activation.

Due to the significant computational efforts necessary to run these experiments, we only have one result per setting, so that comparisons are only indicative. The obtained results demonstrate that structured NODEs with recognition models can identify nonlinear systems from experimental data, i.e., partial and noisy observations.

Open-loop test trajectories One short test trajectory in the trained frequency regime is shown in Fig. 5.14 and illustrates the data fitting capabilities in all three settings and for the linear prior model. The learned models can fit data from a complex nonlinear system excited with different input frequencies, and somewhat generalize to unseen frequencies, as illustrated in Fig. 5.15. The predictions are not perfect, but much more accurate than those of the prior model, as seen in Fig. 5.14–5.15; this is enough to

be used in closed-loop tasks such as control or monitoring. Imposing $\dot{x}_{1/3} = x_{2/4}$ leads to similar performance as without structure, but a physically meaningful state-space representation that can be interpreted in terms of position and velocity. Due to the inaccurate predictions of the prior model, learning its residuals leads to lower performance.

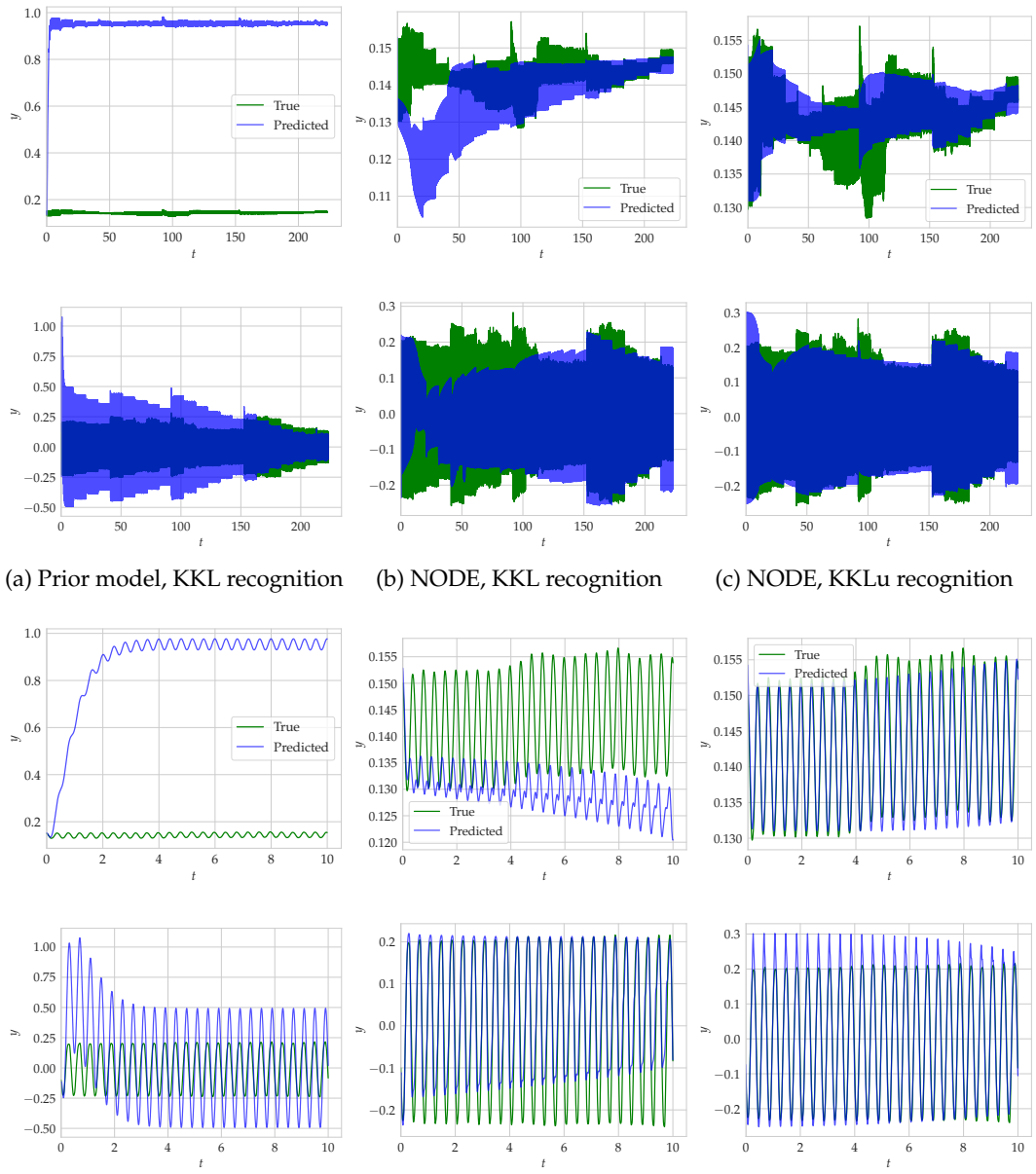
Recognition models With all levels of prior knowledge, the different recognition models lead to comparable results, as illustrated in Table 5.3. For this complex and nonautonomous use case, the direct and RNN+ recognition methods seem easier to train. However, they also take longer due to having more parameters. The KKL and KKLu methods lead to similar performance. To obtain these results, the choice of the gain matrix D was critical. Rigorous analysis of the role of this parameter remains a relevant task for future work [52].

Using the model in an EKF We also evaluate the learned model inside an extended Kalman filter (EKF). The EKF is a classical state estimation tool for nonlinear systems, that takes the measurement and control as input and outputs a probabilistic estimate of the current underlying state. At each time step, it linearizes the dynamics model and output map then proceeds as a linear Kalman filter to estimate the mean and covariance of the current state [174]. In standard applications, the mean of this estimate is then used for control or monitoring of the unmeasured system states. We implement an EKF which receives either $y = h(x) = (x_1, x_4)$ or only $y = h(x) = x_4$ as the measurement, and uses the linear prior or an NODE as dynamics function. In both cases, the NODE estimates are more accurate than those obtained with the linear prior model, as shown in Fig. 5.16–5.17 for a long test trajectory with an input frequency outside of the training regime. As expected, the accuracy for $y = (x_1, x_4)$ is high since we are directly estimating the output. However, the performance difference indicates that the NODE is more accurate and should enable meaningful state estimation for downstream tasks. When $y = x_1$, the EKF using the prior model is off, while it provides reasonable estimates in most frequency regimes when using the learned model.

Full open-loop test trajectory Finally, we also run one long open loop test trajectory over the whole experiment: we estimate the initial state with KKL recognition from the first few measurements, then go over all input frequencies one after the other as they were selected to generate the data. The results, depicted in Fig. 5.18, show that the obtained NODE does not diverge and stays reasonably close to the data even for very long test trajectories in unseen frequency regimes. On the contrary, the linear prior model is far off and stabilizes around the wrong equilibrium point. Interestingly, these long-term predictions are more accurate with KKLu recognition as with KKL, as illustrated in Fig. 5.18.

Conclusion on the experimental dataset

Overall, we find that structured NODEs are able to fit this complex nonlinear dynamical system using experimental data and realistic settings. The predictions of the obtained models are not perfect, but they are much better than those of the prior model, such that they could probably be used to improve feedforward control. This is confirmed by implementing an EKF that uses the learned models for state estimation. Adding structure leads to similar performance, but to a model that can be physically



(a) Prior model, KKL recognition (b) NODE, KKL recognition (c) NODE, KKLu recognition

(d) Zoom on (a) (e) Zoom on (b) (f) Zoom on (c)

Figure 5.18: Test trajectory over the whole experiment with the robotics dataset, using either the linear prior model (left) or the NODE model with $\dot{x}_{1/3} = x_{2/4}$ (middle and right), and KKL (left and middle) or KKLu (right) recognition. A zoom into the beginning of the trajectories is also provided.

interpreted in terms of position and velocity. In the third setting (hard constraints and residual model), the accuracy is lower. This is due to the inaccuracy of the prior model predictions.

5.7 Discussion

The results herein illustrate that observer theory and, in particular, KKL observers are suitable for building recognition models. To the best of our knowledge, this work is the first to propose this connection, hence, it also points to remaining open questions. In particular, the choice of (D, F) plays a role in the performance of KKL observers [52], and methods for tuning them are still needed. Setting (D, F) to HiPPO matrices [30], [161], [162], particular forms of LTI systems that have desirable properties of continuous-time memorization, could be an interesting first step. A better understanding of the performance of recognition models in general would also be of interest. In particular, assessing this performance a priori depending on how observable a region of the state-space is could help generate more informative data, by indicating where in the state-space the observations are best suited for identifying the dynamics.

Incidentally, NODEs can be combined with Convolutional Neural Networks to capture spatial dependencies and learn Partial Differential Equations (PDEs). This is illustrated in [138], [156] and presented explicitly in [175], [176]. In this work, we only consider dynamical systems that can be modeled by ODEs, but expect the proposed approach to extend to PDEs.

5.7.1 Kernel view

On a different note, the NODE formulation for learning the residuals of a prior model (5.11) is similar to kernel Ridge regression [177]. In the most common case, kernel Ridge regression (KRR) is formulated as a nonlinear least-squares regression:

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{N} |y_i - f(x_i)|^2 + \lambda |f|_{\mathcal{H}_k} \tag{5.21}$$

where f is the function to be optimized over a reproducing Hilbert space (RKHS) $\mathcal{H}_k$ with a fixed positive definite kernel k, $|\cdot|_{\mathcal{H}_k}$ is the RKHS norm and (x_i, y_i) are input and output samples. See Sec. 3.3 for a more detailed introduction to kernel methods. This problem formulation has similarities with (5.11), but the term $f(x_i)$ is static in general KRR. For it to correspond to (5.11), we would ideally need $h(x(t_i))$ instead of $f(x_i)$, where x is the solution of an ODE of dynamics $f \in \mathcal{H}_k$. This would boil down to a different KRR formulation with an integral constraint.

Embedding residual model learning into a KRR formulation would be advantageous: if an equivalent of the representer theorem (see Sec. 3.3) could be given for this formulation, then the solution of this problem would be a linear combination of terms $k(x_i, \cdot)$ and thus the search space would be finite-dimensional, enabling efficient computations and theoretical guarantees, which are not given for NODEs.

We can conclude that kernel methods are conceptually well suited for problem (5.11), and there is hope for theoretical guarantees. However, standard results such as KRR cannot be used directly for this problem since the cost function consists in implicit function evaluations, as the solution f only appears in the ODE satisfied by x. To circumvent this, one could consider a discrete model, e.g., the output of a numerical solver for this ODE, and consider each discrete point as a constraint for

the KRR, which could fit into the extension by [178], [179]. Another possibility would be to look for a representer theorem that can take integral constraints into account, as is done for linear differential operators in [178], [179]. We leave this path open for future work.

5.8 Conclusion

As discussed in the previous chapter, in many practical cases, learning an SSM from experimental data involves jointly estimating the state and the dynamics. For systems in the observable canonical form, an HGO can be combined with a GP model and yield theoretical guarantees of joint convergence, as shown in Ch. 4. For more general systems, i.e., with a degree of prior knowledge that can vary from none to enforcing certain constraints or even a parametric model, the general formulation of NODEs is well suited. However, learning physically sound dynamics in realistic settings, i.e., with control inputs and partial, noisy observations, remains challenging. To achieve this, recognition models are needed to efficiently link the observations to the latent state. We show that notions from observer theory can be leveraged to construct such models; for example, KKL observers can filter the information contained in the observations to produce an input of fixed dimension for which a suitable recognition model is guaranteed to exist. We propose to combine recognition models and existing methods for physics-aware NODEs to build a unifying framework, which can learn physically interpretable models in realistic settings. We illustrate the performance of KKL-based recognition in numerical simulations, then demonstrate that the proposed end-to-end framework can learn SSMs from partial observations with an experimental robotics dataset. While these observer-based recognition models are demonstrated in the context of NODEs, they are a separate contribution, which can also be used in various system identification methods; they could be combined with e.g., Neural Controlled Differential Equations [180], Bayesian extensions of NODEs [141], [142], [181] or in general with optimization-based system identification methods [18], [19].

Part II

Machine learning for data-driven observers

In the first part of the thesis, we leverage concepts from nonlinear observer theory to improve dynamics learning from experimental data. We now turn this around and draw on machine learning techniques to enable state estimation. Specifically, the notion of observability itself, i.e., the possibility to reconstruct the latent state from the output, is only simple to assess for linear systems. For nonlinear systems, observability is generally assumed, but it is not uniform over the state-space and cannot easily be analyzed. Therefore, we make use of statistical tools to analyze observability directly from output data in Ch. 6. Also, there are few generic observer designs for nonlinear systems. The KKL observer is one of them, and it has recently been shown that machine learning methods can make it more applicable in practice [116]. However, these numerical KKL observers have free parameters that need to be tuned; we propose an empirical criterion to guide the tuning procedure in Ch. 7.

Chapter 6

Data-driven observability analysis for nonlinear stochastic systems

Résumé La distinguabilité et, par extension, l'observabilité sont des propriétés essentielles des systèmes dynamiques. Établir ces propriétés est complexe, en particulier lorsqu'aucun modèle analytique n'est disponible et qu'elles doivent être déduites directement des données de mesure. La présence de bruit complique encore cette analyse, car les notions standard de distinguabilité sont adaptées aux systèmes déterministes. Nous nous appuyons sur la distinguabilité distributionnelle, qui étend la notion déterministe en comparant les distributions de sorties des systèmes stochastiques. Nous montrons d'abord que les deux concepts sont équivalents pour une classe de systèmes qui inclut les systèmes linéaires. Nous présentons ensuite une méthode pour évaluer et quantifier la distinguabilité distributionnelle à partir des données de sortie. Plus précisément, nous introduisons une quantification mesurant la quantité de données nécessaires pour distinguer deux états initiaux, ce qui induit un spectre continu de distinguabilité. Nous proposons ensuite un *test statistique* pour déterminer un seuil au-dessus duquel deux états peuvent être considérés comme distinguables avec un niveau de confiance élevé. Nous illustrons ces outils en calculant des cartes de distinguabilité sur l'espace d'état en simulation, puis nous tirons parti du test statistique pour comparer différentes configurations de capteurs à partir de données expérimentales.

Abstract Distinguishability and, by extension, observability are key properties of dynamical systems. Establishing these properties is challenging, especially when no analytical model is available and they are to be inferred directly from measurement data. The presence of noise further complicates this analysis, as standard notions of distinguishability are tailored to deterministic systems. We build on distributional distinguishability, which extends the deterministic notion by comparing the distributions of the outputs of stochastic systems. We first show that both concepts are equivalent for a class of systems that includes linear systems. We then present a method to assess and quantify distributional distinguishability from output data. Specifically, we introduce a quantification measuring how much data is required to tell apart two initial states, inducing a continuous spectrum of distinguishability. We then propose a *statistical test* to determine a threshold above which two states can be considered distinguishable with high confidence. We illustrate these tools by computing maps of distinguishability over the state-space in simulation, then leverage the statistical test to compare sensor configurations on hardware.

Parts of this chapter are under review under the title *Data-Driven Observability Analysis for Nonlinear Stochastic Systems* [51]. This work was conducted in collaboration with Pierre-François Massiani from the Institute for Data Science in Mechanical Engineering, RWTH Aachen University. The first two authors contributed equally to this

submission [51] and thus to the results of this chapter. In particular, they jointly developed the main ideas, established the theoretical results, conducted the experiments, and wrote the article.

6.1 Introduction

Distinguishability is the property that allows telling apart different initial states from output measurements; observability then refers to all states being distinguishable. It is a core assumption in observer design, and there exist multiple sufficient criteria to establish observability from analytical models [43]. However, in many cases, such an analytical model is either unavailable or impractical to work with. Examples involve engineering systems with intractable models [5], black-box simulators [10], [23], or systems that are only partially or imperfectly modeled such that the derived properties may not transfer to the real world. In contrast, we often have access to *measurement data* generated by these systems. In this chapter, we propose to infer the observability of the underlying system from such data, without relying on an explicit model of the dynamics.

In many cases, measurements are corrupted by noise. Yet, the standard definition of observability is tailored to deterministic systems [113], and noise is often treated as a perturbation against which an observer should be robust [182]. This perspective circumvents defining stochastic observability by focusing on the robustness of the specific observer. In contrast, we start by defining an inherent notion of observability for *stochastic systems*. We argue that there is a broad class of reasonable definitions, each corresponding to a way of comparing the underlying stochastic processes [183]–[188]. We generalize *distributional distinguishability*, first defined for probabilistic boolean networks (PBNs) [183], to arbitrary nonlinear systems. It consists in comparing the distributions of the output. We show that distributional distinguishability is a reasonable generalization of the deterministic notion by proving that the stochastic and nominal systems share the same classes of indistinguishability under some assumptions, which include linear systems.

Our main contribution is then a method to assess and quantify this distributional distinguishability from measurement data. To achieve this, we propose to use a metric between the distributions of output measurements; its value reflects how different these distributions are. We choose the maximum mean discrepancy (MMD) [106] as this metric, for three reasons. First, it takes values in a continuous set, which induces a continuous spectrum of *relative* distinguishability on which some pairs are more distinguishable than others. Second, it can be estimated from data, hereby establishing distinguishability of initial distributions with high confidence given the data (*absolute* distinguishability) through statistical testing. Third, its continuous value is interpreted as how easily two initial distributions can be told apart, in the sense of how much data the test requires.

We validate the proposed tools in simulation and on hardware. We first demonstrate using the finite-sample MMD to evaluate and quantify distinguishability. We experimentally recover the nominal system's analytical classes of indistinguishability in the cases covered by our theoretical results, and empirically demonstrate how the metric increases as we get farther from a state's class of indistinguishability. This also enables evaluating the influence of the noise on said classes. Further, we illustrate the difference between our test and *empirical Gramians* [189], a tool for data-driven weak observability analysis. This difference mainly resides in the property evaluated: distinguishability, or weak observability. The latter is *local*, while our test recovers a

state's whole class of indistinguishability. We then use the proposed test on a Furuta pendulum for experiment design by checking whether removing specific sensors harms observability. Finally, we begin bridging the gap between the proposed tools and observer design by showcasing how an observer outputs the same distribution over estimated trajectories when the system is initialized in indistinguishable states. This is a first step towards using our tools for a priori error analysis in observer design; the MMD indicates which states observers will fail to distinguish.

This chapter is structured as follows. We first discuss related work in Sec. 6.1.1. We then introduce deterministic and distributional distinguishability in Sec. 6.2, and investigate the relationship between these two notions in specific cases in Sec. 6.2.3. We propose to quantify distributional distinguishability with the MMD in Sec. 6.3, and evaluate the presented tools in Sec. 6.4 before concluding in Sec. 6.5.

6.1.1 Related work

We propose a method to assess and quantify observability from output data without explicitly relying on a dynamics model. To the best of our knowledge, such a model-free, data-driven analysis has never been proposed for nonlinear stochastic systems. We start by reviewing existing notions of stochastic observability and distinguishability and corresponding metrics. Finally, we present an essential concept for the metric we propose: kernel mean embeddings (KMEs).

Stochastic observability

Noise is often treated as a perturbation against which observers should be robust [182]; stochastic observability then reduces to (approximate) convergence of usual observers, possibly up to some probability [190]. Despite its practical appeal, this approach ties observability (a system property) to the observer (a design choice), and neglects the question of how noise affects observability independently of the chosen observer. Answering it requires defining observability for a stochastic system; yet, there is no consensus over such a definition [189].

Many approaches deal with linear systems and rely on the superposition principle to define the observability of the whole system directly [191]. Extensions to nonlinear systems typically result in local notions of observability. A relevant example is stochastic Gramians [189], a generalization of empirical Gramians of nonlinear systems [192], [193]. Similarly to our method, they enable a data-driven analysis of observability with the important difference that they build on *local weak observability* instead of distinguishability, which only provides local information, as we illustrate in Sec. 6.4.

Competing approaches aim at generalizing *distinguishability* [113] of (distributions over) initial states, rather than observability. Core to these approaches is a way to compare the underlying *stochastic processes* induced by the stochastic initial state, dynamics, and measurement. Since stochastic processes can be compared in many more ways than deterministic trajectories, there exist different notions of distinguishability for stochastic systems. Examples involve probabilistic Boolean networks [183]–[185] and other linear systems [186]–[188]; their corresponding definitions of distinguishability are not always explicit, but can be straightforwardly adapted from those of observability. Interestingly, the relationship between such definitions and the existence of a probabilistic observer has not been studied, unlike for deterministic systems [43]. Our approach builds on *distributional distinguishability* [183], and we show that it is equivalent to observability of the nominal system in certain cases, such

as linear dynamics or additive measurement noise. One advantage of this notion is also its amenability to statistical tests provided independent realizations of the system.

Metricizing observability

A practical way to check observability is to summarize it in a scalar value or function. For linear systems, such a metric is given by the rank of the observability matrix or, equivalently, of the observability Gramian [194]. An alternative is the largest singular value of the Gramian, introducing a continuous scale and enabling a *quantification* of observability. These metrics translate locally to nonlinear systems with empirical Gramians [192]; they inform on the weak observability of each state. Gramians and their corresponding metrics have also been extended for stochastic systems [189]. They are formally defined for linear systems with additive or multiplicative noise, and extensions towards arbitrary nonlinear systems are only hinted at [189, Sec. 5]. While our theoretical analysis shares similar restrictions, our approach differs in that:

(i) our definition and quantification are readily applicable to both linear and nonlinear stochastic systems;

(ii) we quantify *distinguishability* of two initial states, instead of weak observability.

Finally, there is a body of research that defines observability [195] and observability metrics [196] based on information-theoretic properties of the output. This definition strongly differs from our approach, which is thus of independent interest.

Kernel mean embeddings

Our quantification of observability relies on KMEs of probability distributions [106]; see Sec. 3.3 for a more formal introduction. In short, these embeddings represent probability distributions in a high-dimensional reproducing kernel Hilbert space (RKHS) where scalar products can be performed efficiently and often estimated from data. Therefrom developed many methods to model and compare random variables, such as modeling of conditional relations [197] or Bayesian inference [198]. Essential for this work is the MMD, a distance between probability distributions that can be estimated from data [199]. The MMD was leveraged in a distribution-free two-sample test able to compare probability distributions from independent samples [111]. We heavily rely on both the MMD and this test, which requires independent samples from the same distributions. This can be problematic for dynamical systems, where the future greatly depends on the past. Some approaches alleviate this problem with mixing [112]; in contrast, we assume independent reinitializations of the system.

6.1.2 Notations

The Borel σ-algebra of an open set $\mathbb{S} \subset \mathbb{R}^n$ is denoted by $\mathcal{B}(\mathbb{S})$, and $\mathcal{M}_1^+(\mathbb{S})$ is the set of probability measures on $(\mathbb{S}, \mathcal{B}(\mathbb{S}))$. We assume open sets $\mathbb{X} \subset \mathbb{R}^{d_x}$ and $\mathbb{Y} \subset \mathbb{R}^{d_y}$, respectively called state and output spaces, and equip them with their Borel σ-algebras. All random variables are defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$. For a random variable Z, the notation $Z \in \mathbb{S}$ means that Z takes values in $(\mathbb{S}, \mathcal{B}(\mathbb{S}))$, and χ_Z is the characteristic function of Z. We denote random variables by capital letters and deterministic quantities by lowercase letters.

6.2 Observability

We first recall a definition of observability for deterministic systems (Sec. 6.2.1), then extend this notion to nonlinear stochastic systems by generalizing distributional distinguishability of PBNs (Sec. 6.2.2). This sets the stage for our first contribution: showing that, for linear dynamics or certain nonlinear systems, the two notions are equivalent (Sec. 6.2.3).

For simplicity, we focus on autonomous systems. All concepts and results translate to control systems by considering distinguishability for a *specific* controller, as is commonly done [113], [192].

6.2.1 Deterministic distinguishability

Consider a deterministic, autonomous system

$$\begin{aligned} x_{t+1} &= f(x_t), \\ y_t &= h(x_t), \end{aligned} \tag{6.1}$$

with arbitrary initial condition $x_0 \in \mathbb{X}$. The function $f : \mathbb{X} \to \mathbb{X}$ is the dynamics, and $h : \mathbb{X} \to \mathbb{Y}$ is the output map. We denote by $\phi(t, x_0) \in \mathbb{X}$ the trajectory of (6.1) initialized in x_0, at time t. Additionally, define the *output trajectory* up to time $T \in \mathbb{N}$ of (6.1) initialized in x_0 as

$$\gamma(T, x_0) = \big(h(\phi(0, x_0)),\ \ldots,\ h(\phi(T, x_0))\big) \in \mathbb{Y}^{T+1}. \tag{6.2}$$

We can then define distinguishability and observability as:

Definition 4 (Sontag [113, Chapter 6])**.** *Let $T \in \mathbb{N}$, and let $x_\mathrm{a}, x_\mathrm{b} \in \mathbb{X}$ be two initial states of (6.1). We say that x_a and x_b are* distinguishable *(in time T) if $\gamma(T, x_\mathrm{a}) \neq \gamma(T, x_\mathrm{b})$. Otherwise, we say that they are* indistinguishable *(in time T). Finally, the system is* observable *(in time T) if all different initial states are distinguishable.*

Intuitively, indistinguishable states produce the same output until time T. It is thus impossible to tell them apart by looking only at the output sequence. In contrast, distinguishability means that the outputs differ after some time.

Definition 4 is binary, meaning that an infinitesimal difference between trajectories can make two states distinguishable. This becomes problematic in practice, in particular for stochastic systems, where two sample paths from the *same* initial point will differ almost surely for nontrivial noise, dynamics, and measurement.

6.2.2 Distributional distinguishability

We therefore need a generalized notion of distinguishability to handle stochastic systems. They come with the additional hurdle that output measurements are now random variables (RVs). Hence, the question of distinguishability pertains to the more general problem of how to compare stochastic processes. There are many suitable ways to define the "equivalence" of stochastic processes, each leading to a specific notion of stochastic observability [183], [185]. We focus on distributional distinguishability, since the similarity with the deterministic case makes it a natural extension and because, as we will show, it lends itself to data-driven evaluation.

Consider a stochastic dynamical system of the form

$$\begin{aligned} X_{t+1} &= F(X_t, \eta_t), \\ Y_t &= H(X_t, \epsilon_t), \end{aligned} \tag{6.3}$$

The state X_t and output Y_t are now RVs taking values in $\mathbb{X}$ and $\mathbb{Y}$, resp. . The RV $\eta_t \in \mathbb{R}^{d_\eta}$ represents the stochastic part of the dynamics, and $\epsilon_t \in \mathbb{R}^{d_\epsilon}$ the one of the measurement. We assume that (6.3) is a noisy version of (6.1), that is, $F(x, 0) = f(x)$ and $H(x, 0) = h(x)$. If F is linear, i.e., $F(x, n) = A \cdot x + Q \cdot n$, we recover the classical case of a linear system with additive process noise. If, additionally, $(\eta_t)_t$ is a family of Gaussian, independent variables, then $(X_t)_t$ is a discrete-time Ornstein-Uhlenbeck process.

We allow stochastic initializations, meaning that there exists an initial distribution μ defined on $\mathbb{X}$ such that $X_0 \sim \mu$. We assume that η_t and ϵ_t are independent of the past states $(X_s)_{s \leq t}$. We do not make further assumptions on the noise; e.g., we allow non-centered or autocorrelated noise. Finally, we denote by $\Phi(t, \mu)$ the random trajectory of (6.3) initialized as per μ, at time $t \in \mathbb{N}$, and define

$$\Gamma(T, \mu) = \big(H(\Phi(0, \mu), \epsilon_0),\ \ldots,\ H(\Phi(T, \mu), \epsilon_T)\big) \in \mathbb{Y}^{T+1}, \tag{6.4}$$

the random output trajectory up to time $T \in \mathbb{N}$. We denote its law by $\mathbb{P}^T_\mu$. We are now ready to generalize distributional distinguishability, first introduced for PBNs [183], as follows:

Definition 5 (Distributional distinguishability)**.** *Let $T \in \mathbb{N}$, and let $\mu_\mathrm{a}, \mu_\mathrm{b}$ be two initial distributions of (6.3). We say that μ_a and μ_b are* distributionally distinguishable *(in time T) if $\mathbb{P}^T_{\mu_\mathrm{a}} \neq \mathbb{P}^T_{\mu_\mathrm{b}}$. Otherwise, we say that they are* distributionally indistinguishable *(in time T). Finally, the system is* distributionally observable *(in time T) if all different initial distributions are distributionally distinguishable.*

Distributional observability thus compares the laws of the random trajectories instead of comparing the trajectories themselves, as Definition 4 does. Intuitively, it focuses on whether one can tell apart different initial distributions by *repeating the experiment* multiple times.

Since distributional distinguishability is defined for arbitrary initial distributions, it is also meaningful to talk about distributional distinguishability of *initial states*; the initial distributions involved are then accordingly-centered Diracs. In this case, we abuse notations and use x directly instead of δ_x in $\Phi(t, \cdot), \Gamma(T, \cdot)$, and $\mathbb{P}^T_\cdot$.

6.2.3 Noise and distinguishability

Distributional observability reduces to the classical, deterministic notion when the system is deterministic with deterministic initializations; the distributions of $\Phi(T, x_\mathrm{a})$ and $\Phi(T, x_\mathrm{b})$ are then Diracs and are thus equal iff. the variables are equal almost surely. Therefore, we study whether noise influences distinguishability. In particular, can introducing noise make two states distinguishable? Conversely, can noise hide the difference between nominally-distinguishable initial states? In general, yes; the classes of indistinguishability of the nominal and noisy systems differ, as we illustrate in Sec. 6.4. In this section, we study conditions under which the noise does *not* influence distinguishability. We start with measurement noise and show that, under a technical assumption, it does not affect distinguishability. This implies that the deterministic and stochastic notions coincide in the absence of process noise. We then treat linear systems, for which this equivalence always holds.

Measurement noise

We start with the following assumption to restrict the influence of measurement noise to the output of h:

Assumption 6. *There exists a measurable function $K : \mathbb{Y} \times \mathbb{R}^{d_\epsilon} \to \mathbb{Y}$ such that $H(x, e) = K(h(x), e)$.*

This enables introducing the output before its corruption by measurement noise, an RV denoted by $\hat{Y}_t = h(X_t)$ independent of ϵ_t.

Assumption 7. *For all $\nu_\mathrm{a}, \nu_\mathrm{b} \in \mathcal{M}_1^+(\mathbb{Y}^{T+1})$ with $\nu_\mathrm{a} \neq \nu_\mathrm{b}$, there exists $A \in \mathcal{B}(\mathbb{Y}^{T+1})$ such that, by applying K element-wise,*

$$\int_{\mathbb{Y}^{T+1}} \mathbb{P}[K(\hat{\gamma}, \epsilon) \in A] \mathrm{d}\nu_\mathrm{a}(\hat{\gamma}) \neq \int_{\mathbb{Y}^{T+1}} \mathbb{P}[K(\hat{\gamma}, \epsilon) \in A] \mathrm{d}\nu_\mathrm{b}(\hat{\gamma}). \tag{6.5}$$

This assumption is a *generalized injectivity condition*: it requires the mapping from a distribution ν of $\hat{\Gamma}$ to the output distribution after corruption by measurement noise to be injective, where $\hat{\Gamma} = (\hat{Y}_0, \dots, \hat{Y}_T)$ is the output trajectory *before* the measurement noise. It ensures that if two distributions of $\hat{\Gamma}$ are distinct, they remain different after applying measurement noise.

Example 1 (Additive noise)**.** *Assumption 7 is satisfied for additive noise. Indeed, assume $K(\hat{\gamma}, e) = \hat{\gamma} + e$, and take $\nu_\mathrm{a}, \nu_\mathrm{b} \in \mathcal{M}_1^+(\mathbb{Y}^{T+1})$ two distinct probability measures. Let $\hat{\Gamma}_\mathrm{a} \sim \nu_\mathrm{a}$, $\hat{\Gamma}_\mathrm{b} \sim \nu_\mathrm{b}$, and assume that $\hat{\Gamma}_\mathrm{a}, \hat{\Gamma}_\mathrm{b}$, and ϵ are independent. By definition, we have for all $A \in \mathcal{B}(\mathbb{Y}^{T+1})$:*

$$\int_{\mathbb{Y}^{T+1}} \mathbb{P}[K(\hat{\gamma}, \epsilon) \in A] \mathrm{d}\nu_i(\hat{\gamma}) = \mathbb{P}[K(\hat{\Gamma}_i, \epsilon) \in A], \tag{6.6}$$

where $i \in \{a, b\}$. Therefore, Assumption 7 is satisfied iff. $\hat{\Gamma}_\mathrm{a} + \epsilon$ and $\hat{\Gamma}_\mathrm{b} + \epsilon$ have different laws, iff. $\chi_{\hat{\Gamma}_\mathrm{a}+\epsilon} \neq \chi_{\hat{\Gamma}_\mathrm{b}+\epsilon}$. By independence of the noise, $\chi_{\hat{\Gamma}_i+\epsilon} = \chi_{\hat{\Gamma}_i} \cdot \chi_\epsilon$. Therefore, Assumption 7 holds iff. $\chi_{\hat{\Gamma}_\mathrm{a}} \neq \chi_{\hat{\Gamma}_\mathrm{b}}$, which is true since $\nu_\mathrm{a} \neq \nu_\mathrm{b}$ by assumption.

Importantly, Assumption 7 enables decoupling the effect of measurement and process noise on distributional distinguishability; measurement noise then does not affect distinguishability.

Theorem 5. *Let $\mu_\mathrm{a}, \mu_\mathrm{b} \in \mathcal{M}_1^+(\mathbb{X})$ be two initial distributions, and let Assumption 6 hold. If μ_a and μ_b are distributionally distinguishable, the distributions of the corresponding output trajectories before corruption by measurement noise $\hat{\Gamma}_\mathrm{a} \doteq (h(\Phi(t, \mu_\mathrm{a}))_{t=0}^T$ and $\hat{\Gamma}_\mathrm{b} \doteq (h(\Phi(t, \mu_\mathrm{b}))_{t=0}^T$ differ. Under Assumption 7, the converse is also true.*

Proof. Let $\mu_\mathrm{a}, \mu_\mathrm{b} \in \mathcal{M}_1^+(\mathbb{X})$ be distinguishable, and take $A \in \mathcal{B}(\mathbb{Y}^{T+1})$ such that $\mathbb{P}_{\mu_\mathrm{a}}^T[A] \neq \mathbb{P}_{\mu_\mathrm{b}}^T[A]$. Let $i \in \{a, b\}$, and ν_i be the distribution of $\hat{\Gamma}_i$. Marginalizing $\Gamma(T, \mu_i)$ the noisy output trajectory on its counterpart without measurement noise $\hat{\Gamma}_i$ yields

$$\mathbb{P}_{\mu_i}^T[A] = \mathbb{P}\left[\Gamma(T, \mu_\mathrm{a}) \in A\right] = \int_{\mathbb{Y}^{T+1}} \mathbb{P}\left[K(\hat{\gamma}, \epsilon) \in A\right] \mathrm{d}\nu_i(\hat{\gamma}), \tag{6.7}$$

where $\mathbb{P}\left[K(\hat{\gamma}, \epsilon) \in A\right]$ is the probability of the observation lying in A given that the nominal observation is $\hat{\gamma}$. It is thus independent of μ_a and μ_b, as it only depends on the measurement noise. Since $\mathbb{P}_{\mu_\mathrm{a}}^T[A] \neq \mathbb{P}_{\mu_\mathrm{b}}^T[A]$, (6.7) implies that the integral of $\mathbb{P}\left[K(\cdot, \epsilon) \in A\right]$ differs when considered w.r.t. ν_a and ν_b. Therefore, $\nu_\mathrm{a} \neq \nu_\mathrm{b}$, yielding

the first claim. Conversely, assume that ν_a and ν_b the distributions of the output trajectories without measurement noise $\hat{\Gamma}_a$ resp. $\hat{\Gamma}_b$ differ. Assumption 7 ensures the existence of $A \in \mathcal{B}(\mathbb{Y}^{T+1})$ such that

$$\int_{\mathbb{Y}^{T+1}} \mathbb{P}[K(\hat{\gamma}, \epsilon) \in A] \mathrm{d}\nu_a(\hat{\gamma}) \neq \int_{\mathbb{Y}^{T+1}} \mathbb{P}[K(\hat{\gamma}, \epsilon) \in A] \mathrm{d}\nu_b(\hat{\gamma}). \tag{6.8}$$

By (6.7), this shows $\mathbb{P}^T_{\mu_a} \neq \mathbb{P}^T_{\mu_b}$, hence μ_a, μ_b are distributionally distinguishable. This concludes the proof. □

Corollary 1. *Let Assumptions 6 and 7 hold. Assume that there is no process noise, i.e., $\eta_t = 0$ almost surely for all t. Let $x_a, x_b \in \mathbb{X}$. Then, x_a and x_b are distinguishable for (6.1) iff. they are distributionally distinguishable for (6.3).*

Proof. By Theorem 5, x_a and x_b are distributionally distinguishable iff. the corresponding output trajectories without measurement noise $\hat{\Gamma}_a$ and $\hat{\Gamma}_b$ have different distributions. From $\eta = 0$ almost surely, we have $\forall\, x \in \mathbb{X}, (h(\Phi(t, x))_{t=0}^T = \gamma(T, x)$ almost surely: the trajectory without measurement noise and the nominal trajectory are equal almost surely. Hence, x_a and x_b are distributionally distinguishable iff. the corresponding deterministic trajectories differ, which is the definition of distinguishability as per Definition 4. □

Unfortunately, this corollary does not generalize to non-zero process noise, as we illustrate on an example in Sec. 6.4. We expect that such a generalization requires strong joint assumptions on the noise and dynamics, in the absence of which the trajectories of the nominal and noisy systems need not correlate.

Linear systems

For linear systems, however, the superposition principle enables such a generalization.

Theorem 6. *Assume that (6.3) is linear, i.e., $F(x, n) = Ax + Qn$ and $H(x, e) = Cx + Re$. Then, two initial states $x_a, x_b \in \mathbb{X}$ are distributionally distinguishable iff. they are distinguishable for (6.1). Additionally, (6.3) is distributionally observable iff. (6.1) is observable.*

This guarantees that, for linear systems, *noise preserves the classes of indistinguishability* under the single assumption that the noise is independent of the past trajectory. Additionally, Kalman's criterion on the observability matrix enables checking distributional observability. We emphasize that the second claim is *not* a consequence of the first one; distributional observability requires all initial distributions to be distinguishable rather than initial states. This statement thus means that, for a linear system, distinguishability of all initial states ensures observability; there is no gain in considering initial distributions.

Proof. For $\mu \in \mathcal{M}_1^+(\mathbb{X})$, the superposition principle holds:

$$\Gamma(T, \mu) = \Gamma(T, 0) + \gamma(T, \mu), \tag{6.9}$$

where we abusively denote $\gamma(T, X_0)$ with $X_0 \sim \mu$ by $\gamma(T, \mu)$. In other words, the stochastic output trajectory is composed of the trajectory initialized at zero, whose randomness is only due to the process and measurement noise, and the deterministic trajectory initialized according to μ, whose randomness is only due to the random

initialization. Crucially, $\Gamma(T,0)$ and $\gamma(T,\mu)$ are thus independent. Hence, we have $\chi_{\Gamma(T,\mu)} = \chi_{\Gamma(T,0)} \cdot \chi_{\gamma(T,\mu)}$.

We now move on to the main proof. Let $x_\mathrm{a}, x_\mathrm{b} \in \mathbb{X}$. We have the following equivalences: they are distributionally indistinguishable iff. $\mathbb{P}^T_{x_\mathrm{a}} = \mathbb{P}^T_{x_\mathrm{b}}$, iff. $\chi_{\Gamma(T,x_\mathrm{a})} = \chi_{\Gamma(T,x_\mathrm{b})}$, iff. $\chi_{\gamma(T,\delta_{x_\mathrm{a}})} = \chi_{\gamma(T,\delta_{x_\mathrm{b}})}$, iff. $\gamma(T,x_\mathrm{a}) = \gamma(T,x_\mathrm{b})$. This shows that x_a and x_b are distributionally indistinguishable iff. they are deterministically indistinguishable.

From this, we immediately have that distributional observability implies deterministic observability: if all pairs of initial states are distributionally distinguishable, then they are deterministically distinguishable, thus the nominal system is observable. To complete the proof of the second statement, we now show that the converse is also true. Assume deterministic observability, and let $\mu_\mathrm{a}, \mu_\mathrm{b} \in \mathcal{M}_1^+(\mathbb{X})$ be two initial distributions with $\mathbb{P}^T_{\mu_\mathrm{a}} = \mathbb{P}^T_{\mu_\mathrm{b}}$. We show $\mu_\mathrm{a} = \mu_\mathrm{b}$. A similar reasoning as previously shows $\chi_{\gamma(T,\mu_\mathrm{a})} = \chi_{\gamma(T,\mu_\mathrm{b})}$, i.e., $\gamma(T,\mu_\mathrm{a})$ and $\gamma(T,\mu_\mathrm{b})$ have the same law and, thus, μ_a and μ_b coincide on all sets A in the σ-algebra $\mathcal{S} = \gamma^{-1}(T, \mathcal{B}(\mathbb{Y}^{T+1})) \subset \mathcal{B}(\mathbb{X})$. We show $\mathcal{S} = \mathcal{B}(\mathbb{X})$. From observability of (6.1), $\gamma(T,\cdot)$ is injective. Since it is also linear, it admits a continuous (and thus, measurable) left inverse, say, $\lambda : \mathbb{Y}^{T+1} \to \mathbb{X}$. We are now ready to conclude. Take $A \in \mathcal{B}(\mathbb{X})$, and define $B = \lambda^{-1}(A) \in \mathcal{B}(\mathbb{Y}^{T+1})$. We have, by definition of λ as a left inverse of γ, $\gamma^{-1}(T,B) = \gamma^{-1}(T,\lambda^{-1}(A)) = A$. This shows that $A \in \mathcal{S}$, i.e., that $\mathcal{B}(\mathbb{X}) = \mathcal{S}$. Therefore, $\mu_\mathrm{a}(A) = \mu_\mathrm{b}(A)$ for all $A \in \mathcal{B}(\mathbb{X})$, which shows distributional observability of (6.3). □

Albeit distributional and deterministic distinguishability differ in general, it is reasonable to expect that a small amount of noise only mildly affects the distinguishability of two states. In the next section, we propose a *quantification* of distinguishability to formalize the above intuition, and a method to estimate it from output data.

6.3 Measuring distributional observability

Distributional observability is defined as the equality of certain probability distributions. We propose to extend this binary notion to a continuous spectrum of relative distinguishability by quantifying it with the MMD, a distance between said distributions measuring how easily they can be told apart. Additionally, the MMD can be estimated from data. This leads to our main methodological contribution: use output measurements to:

(i) approximate relative distinguishability;

(ii) leverage a two-sample test [111] to assess distributional observability with high confidence from output measurements.

6.3.1 The MMD to measure relative distinguishability

Background

The background necessary to define the MMD and state the results that justify our method is provided in Sec. 3.3. In particular, we denote the KME of the distribution $\mathbb{P}$ by $\mu_\mathbb{P}$. In general, the MMD is only a semi-metric, since the embedding $\mathbb{P} \mapsto \mu_\mathbb{P}$ may not be injective. Kernels for which it *is* injective are called *characteristic* [200]; they include the squared exponential kernel. From now on, we assume a characteristic kernel.

Remark 13 (A kernel on trajectories). *We emphasize that the considered kernel k is defined on the output trajectory space $\mathbb{Y}^{T+1}$, and not on the state-space $\mathbb{X}$ nor on the output space $\mathbb{Y}$. Alternatively to defining a kernel on $\mathbb{Y}^{T+1}$ directly, one can construct it from a base kernel ℓ defined on $\mathbb{Y}$ by taking $k = \bigotimes_{t=0}^{T} \ell$ [200], where $\bigotimes$ denotes the tensor product. For instance, the expression of a squared exponential kernel on $\mathbb{Y}^{T+1}$ with scalar width $\sigma > 0$, which is known to be characteristic [106, Table 3.1], is*

$$k(\gamma_{\mathrm{a}}, \gamma_{\mathrm{b}}) = s^2 \exp\left[-\frac{1}{2\sigma^2} \sum_{t=0}^{T} \sum_{i=1}^{d_y} (\gamma_{\mathrm{a},i,t} - \gamma_{\mathrm{b},i,t})^2\right], \tag{6.10}$$

where $\gamma_{\mathrm{a}}, \gamma_{\mathrm{b}} \in \mathbb{R}^{(T+1)\times d_y}$ are output trajectories. In this chapter, we fix the scaling parameter $s = 1$.

A metric of distinguishability

The value of the MMD thus indicates whether initial distributions are distinguishable.

Proposition 1. *Let $\mu_{\mathrm{a}}, \mu_{\mathrm{b}}$ be two initial distributions. Assume that the kernel k is characteristic. Then, $\mathrm{MMD}[\mathbb{P}^T_{\mu_{\mathrm{a}}}, \mathbb{P}^T_{\mu_{\mathrm{b}}}] = 0$ iff. μ_{a} and μ_{b} are distributionally indistinguishable.*

Proof. The MMD is a metric when the kernel is characteristic [200], which implies that $\mathrm{MMD}^2[\mathbb{P}^T_{\mu_{\mathrm{a}}}, \mathbb{P}^T_{\mu_{\mathrm{b}}}] = 0$ iff. $\mathbb{P}^T_{\mu_{\mathrm{a}}} = \mathbb{P}^T_{\mu_{\mathrm{b}}}$. The result then follows from Definition 5. □

The information the MMD carries is richer than a simple yes-or-no indication. Indeed, it takes continuous values; low values reveal similar distributions. Therefore, the MMD extends the binary notion of *absolute* distinguishability to a continuous spectrum of *relative* distinguishability, on which some pairs of initial distributions are more (or less) distinguishable than others. We postpone further interpreting this value to Sec. 6.3.2.

6.3.2 Finite-sample approximation

To access both the relative and absolute notions, one needs to evaluate the MMD. We rely on a finite-sample approximation [111] for this and use a two-sample test to conclude on absolute distinguishability, since the approximation is never exactly 0. Finally, this test helps refine the interpretation of the MMD value.

Background – finite-sample MMD

While a closed-form formula is often unattainable or requires perfect system knowledge, the MMD can be approximated from independent samples of output trajectories by the following empirical estimator.

Theorem 7 (Gretton, Borgwardt, Rasch, *et al.* [111]). *Let $\mu_{\mathrm{a}}, \mu_{\mathrm{b}}$ be two initial distributions. Let $\Gamma_{\mathrm{a},1}, \ldots, \Gamma_{\mathrm{a},m}$ be independent copies of $\Gamma(\mu_{\mathrm{a}}, T)$, and $\Gamma_{\mathrm{b},1}, \ldots, \Gamma_{\mathrm{b},n}$ independent copies of $\Gamma(\mu_{\mathrm{b}}, T)$. Assume that k is bounded, i.e., $0 \leq k \leq K$ for some $K \in \mathbb{R}$. Then,*

$$\mathrm{MMD}_b^2[m, n] = \frac{1}{m^2} \sum_{i,j=1}^{m} k(\Gamma_{\mathrm{a},i}, \Gamma_{\mathrm{a},j}) + \frac{1}{n^2} \sum_{i,j=1}^{n} k(\Gamma_{\mathrm{b},i}, \Gamma_{\mathrm{b},j}) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(\Gamma_{\mathrm{a},i}, \Gamma_{\mathrm{b},j}) \tag{6.11}$$

converges in probability to $\mathrm{MMD}^2[\mathbb{P}^T_{\mu_\mathrm{a}}, \mathbb{P}^T_{\mu_\mathrm{b}}]$ *with the following concentration bound for* $\epsilon > 0$*:*

$$\mathbb{P}\left[\left|\mathrm{MMD}_b[m,n] - \mathrm{MMD}[\mathbb{P}^T_{\mu_\mathrm{a}}, \mathbb{P}^T_{\mu_\mathrm{b}}]\right| > 2\left(\sqrt{\frac{K}{m}} + \sqrt{\frac{K}{n}}\right) + \epsilon\right] \leq 2\exp\left[-\frac{\epsilon^2}{2K}\cdot\frac{mn}{m+n}\right]. \tag{6.12}$$

The outcome of (6.11) is a noisy, biased, and finite-sample approximation of the population value $\mathrm{MMD}[\mathbb{P}^T_\mu, \mathbb{P}^T_\nu]$.

In practice, we quantify the distinguishability of two initial distributions μ_a and μ_b using two datasets of output trajectories. We initialize the system multiple times as per μ_a resp. μ_b and obtain $D_{\mu_\mathrm{a}} = \{\gamma_{\mathrm{a},i,t}, t \in \{0,\ldots,T\}, i \in \{1,\ldots,m\}\}$ and $D_{\mu_\mathrm{b}} = \{\gamma_{\mathrm{b},i,t}, t \in \{0,\ldots,T\}, i \in \{1,\ldots,n\}\}$, where $\gamma_{\mathrm{a},i,t}$ resp. $\gamma_{\mathrm{b},i,t}$ is the realization of $\Gamma_{\mathrm{a},i}$ resp. $\Gamma_{\mathrm{b},i}$ at time t. We then compute the estimator (6.11) on these two datasets. We can rely on the value of the estimator to approximate relative distinguishability, however, we cannot use it directly to check absolute distinguishability. Instead, we use a statistical test.

A two-sample test of distinguishability

To determine whether two initial distributions μ_a and μ_b are distinguishable from the value of 6.11, we perform a statistical test of the null hypothesis

$$H_0 : \mathbb{P}^T_{\mu_\mathrm{a}} = \mathbb{P}^T_{\mu_\mathrm{b}} \quad \text{vs.} \quad H_1 : \mathbb{P}^T_{\mu_\mathrm{a}} \neq \mathbb{P}^T_{\mu_\mathrm{b}}. \tag{6.13}$$

Such a test checking for the equality of two distributions based on samples is called a *two-sample test*. We propose to use one based on the concentration bound (6.12) [111]. One specifies an acceptable Type I risk $\alpha \in (0,1)$, and the test provides an *acceptance region threshold*; if the outcome of (6.11) falls outside of that region, one rejects the null hypothesis H_0 with confidence level at least $1-\alpha$.

Proposition 2. *Let* $\alpha \in (0,1)$*. Under the setting of Theorem 7 with* $n = m$*, consider a realization of* $\mathrm{MMD}_b[m,m]$*. If this empirical estimate is such that*

$$\mathrm{MMD}_b[m,m] \geq \sqrt{\frac{2K}{m}}\left(1 + \sqrt{2\ln\alpha^{-1}}\right) \doteq \kappa, \tag{6.14}$$

then μ_a *and* μ_b *are distributionally distinguishable in time* T *with confidence level* $1-\alpha$*.*

Proof. This immediately follows from [111, Corollary 9]. □

The threshold κ is conservative for many practical applications. It is common to use *bootstrapping* instead, as suggested in [111] and implemented in their original code. Bootstrapping [201] is a well-known statistical technique to estimate a population value by artificially resampling a given set of samples. Typically, one can draw a new such set n times by sampling with replacement from the original samples each time. Then, the population value can be computed for these n sets of samples, and its histogram or other aspects of this empirical distribution can be studied. In particular, for the two-sample test of Prop. (2), one considers the aggregated data, i.e., the concatenation of D_{μ_a} and D_{μ_b} described above. One resamples this aggregated dataset n times by shuffling the samples, which is equivalent to resampling the labels of each realization. The population value to compute is then the MMD between the

first and the last m samples of the concatenated dataset. This value is computed $n = 1000$ times after resampling, and κ is set to the $(1-\alpha)$-quantile of these MMD values. We use this bootstrapped value described in [111] for κ in Sec. 6.4.

From qualitative properties to sample bounds

In Sec. 6.3.1, we interpreted the MMD as a measure of distinguishability based on the argument that it takes values in a continuous set. The two-sample test of Proposition 2 allows us to refine this interpretation: the value of the MMD indicates how much data is required for the test to successfully reject the null hypothesis H_0 with high confidence. In other words, the MMD provides a sample bound for the test to achieve a specific Type II error $\beta \in (0,1)$[1].

Corollary 2. *Let $\beta \in (0,1)$. Under the setting of Theorem 7 with $n = m$, assume that the null hypothesis is incorrect and let $z = \mathrm{MMD}[\mathbb{P}_\mu^T, \mathbb{P}_\nu^T] > 0$. Assume that*

$$m > \frac{K}{z^2}\left(4 + \sqrt{2} + 2\left(\sqrt{\ln\frac{1}{\beta}} + \sqrt{\ln\frac{2}{\beta}}\right)\right)^2. \tag{6.15}$$

Then, the null hypothesis is rejected with probability at least $1-\beta$.

Proof. We first denote $\mathrm{MMD}_b[m,m]$ by x, and introduce $A_m = \{x < \kappa\}$ the test's acceptance region of level β and $B_m = \{|x - z| \leq \delta\}$ the high-probability region of the concentration bound (6.12), with

$$\epsilon = 2\sqrt{\frac{K}{m}\ln\frac{2}{\beta}}, \qquad \delta = 2\sqrt{\frac{K}{m}}\left(2 + \sqrt{\ln\frac{2}{\beta}}\right). \tag{6.16}$$

H_0 is rejected if $A_m \cap B_m = \emptyset$, i.e., if it is not possible for the empirical MMD estimate to be both under the rejection threshold κ and in the high-probability region close to the true MMD value z. Notice that $\kappa < z - \delta$ implies that if $x < \kappa$, then $|z - x| > \delta$. Thus, if we pick $x \in A_m$, then $|z - x| > \delta$, i.e., $x \in B_m^{\mathrm{C}}$: $A_m \subseteq B_m^{\mathrm{C}}$. This is equivalent to $A_m \cap B_m = \emptyset$ and yields $\mathbb{P}[A_m] \leq \mathbb{P}[B_m^{\mathrm{C}}]$. Hence, it is sufficient for H_0 to be rejected with probability at least $1-\beta$ to have:

(i) $\kappa < z - \delta$;

(ii) $\mathbb{P}[B_m^{\mathrm{C}}] \leq \beta$.

Replacing the value of ϵ chosen above into (6.12) yields (ii). Then, replacing the value of κ in (6.14) and the value of δ above into (i) yields (6.15). Therefore, the bound (6.15) guarantees (i) and (ii), which is sufficient to ensure that H_0 is rejected with probability at least $1-\beta$. □

Albeit the threshold (6.15) depends on z and thus cannot be used in practice to guide data collection, it sheds light on the interpretation of the MMD by translating a qualitative property (it measures how different outputs from two initial distributions are) into practical considerations (how much data is required to tell them apart). Intuitively, initial distributions leading to a high MMD can be told apart with confidence given only a small number of output trajectories.

[1] As discussed in [111], it is impossible to find such a bound without assumptions on the distributions at hand. In what follows, the assumption consists in assuming that $z = \mathrm{MMD}[\mathbb{P}_\mu^T, \mathbb{P}_\nu^T] > 0$ is known.

6.4 Experimental results

The proposed tools enable a data-driven observability analysis of nonlinear stochastic systems. We first verify that adding noise preserves distributional observability for linear systems, as stated in Theorem 6. We then illustrate the real-valued quantification by computing it on the whole state-space for a nonlinear Duffing oscillator, revealing a continuous increase as we get away from the class of indistinguishability. This also enables measuring the effects of both measurement and process noise on said classes, illustrating the results of Sec. 6.2.3. Finally, we demonstrate on a Furuta pendulum how to test whether a sensor configuration distinguishes certain states, with implications on experiment design[2].

Remark 14. *We have focused on discrete-time systems; yet, dynamical systems are often described in continuous time and only the sensor measurements are discrete-time. This discrepancy is not problematic: first, the results of Sec. 6.3 still hold if the state is a continuous-time process, even though we have not considered it for clarity. Second, and more practically, most of the following examples are stochastic differential equations (SDEs) solved numerically with a fixed time-step (using [202]), and are thus effectively discrete-time processes. Therefore, the proposed methodology applies seamlessly to discrete-time and continuous-time systems, and we use both in the following sections.*

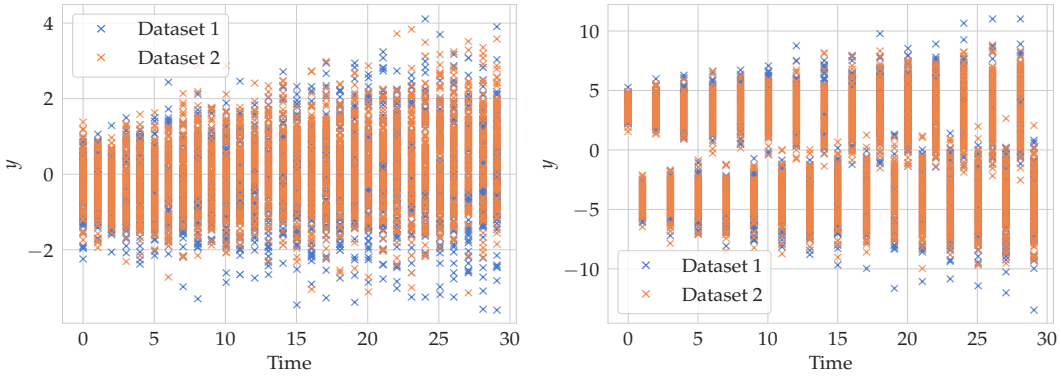
In the following, we use $\alpha = 0.05$ and a standard squared exponential kernel (6.10). A common heuristic to select the width parameter σ is to pick the median pairwise distance between the data sets at hand [111]. It is not directly applicable here, however: our interpretation of the MMD as a measure of relative distinguishability (Sec. 6.3.1) requires that σ be the same for all points to be compared. Indeed, comparing different values of MMD is only meaningful if the underlying embeddings are all in the *same* RKHS, that is, use the same kernel function and parameter values. Therefore, we suggest the following meta-heuristic. We start by computing the heuristic by [111] for each pair of datasets of interest. Then, we set σ as the 0.1-quantile of all previous values. These low values correspond to distributions that are indistinguishable, or hardly distinguishable, so σ has the correct order of magnitude for the noise of the system. We run all experiments with the fixed value of σ given by this meta-heuristic; the exact value is given in each section. This strategy leads to satisfying results in all our use cases, however, it is but an arbitrary choice; one should ensure σ is set to a suitable order of magnitude, otherwise the problem might be badly conditioned and all MMD values close to 0.

6.4.1 Case of linear systems: illustration of Theorem 6

Theorem 6 states that, for linear systems, the presence of noise does not alter distinguishability between two states and, by extension, observability. We illustrate this numerically in the following sections. We first check that the absolute distinguishability of a linear stochastic system as determined by our statistical test from noisy output data is consistent with the distinguishability of the nominal system. We then showcase that the empirical class of indistinguishability recovered by the test for the stochastic system is indeed the same as the analytical class of indistinguishability of the nominal system.

Remark 15. *For a deterministic, linear time-invariant (LTI) system, be it continuous-time or discrete-time, we have $y_t = Cx_t$. Kalman's observability criterion [113, Sec. 6.2] for LTI*

[2]Code to reproduce the results is available at github.com/PFMassiani/data-obs.git.



(a) LTI system (6.20)

(b) LTI system (6.21)

Figure 6.1: Data collected from two linear stochastic systems with process and measurement noise. The proposed statistical test evaluates the left dataset as distinguishable, but not the right one; this is confirmed by the deterministic observability of these linear systems, but is not obvious from looking at the measurements.

systems states that the system is observable iff. the observability matrix

$$O = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{d_x-1} \end{pmatrix} \tag{6.17}$$

has rank d_x, where d_x is the dimension of the state and A is the state transition matrix. The unobservable subspace of the system is then $\ker(O)$. *In particular, the* class of indistinguishability *of any initial state x_a is the set of states from which it is indistinguishable; this class is straightforwardly*

$$x_\mathrm{a} + \bigcap_{i=0}^{d_x-1} (CA^i)^\perp. \tag{6.18}$$

Distributional and deterministic distinguishability

We first show that two distinguishable (resp. indistinguishable) states for the nominal system are indeed identified as distributionally distinguishable (resp. indistinguishable) by our statistical test. We illustrate this numerically on two discrete-time stochastic LTI systems of the form:

$$\begin{aligned} X_{t+1} &= AX_t + QW_t, \\ Y_t &= CX_t + \epsilon_t, \end{aligned} \tag{6.19}$$

first a discretized model of a harmonic oscillator with

$$A = \begin{pmatrix} 1 & 0.01 \\ -0.01 & 1 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top, \tag{6.20}$$

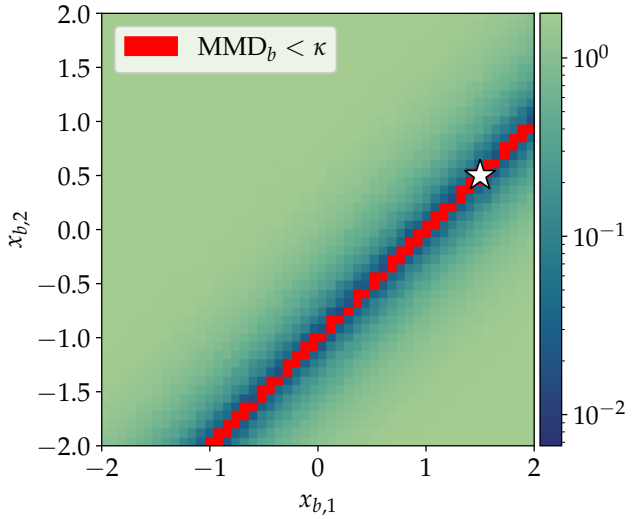


Figure 6.2: MMD over a grid for linear system (6.22) and reference point $x_a = (1.5, 0.5)$ (white star). The empirical class of indistinguishability (red points) is computed from output data from the noisy system. It matches the analytical class of indistinguishability of the nominal system.

then the following system with

$$A = \begin{pmatrix} 0.2 & -0.4 \\ -0.6 & -0.8 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 2 \end{pmatrix}^\top, \tag{6.21}$$

where $W_t \sim \mathcal{N}(0, 0.1)$ is Gaussian process noise, $Q = 0.01\mathbb{I}_2$, and $\epsilon_t \sim \mathcal{N}(0, 0.1)$ is Gaussian measurement noise. We collect a set of 100 output trajectories of length 30 s from Gaussian initial distributions of covariance $0.001\mathbb{I}_2$ and means $x_a = (0, 1)$ and $x_b = (0, 2)$ for (6.20), resp. $x_a = (0, 1)$ and $x_b = (2, 0)$ for (6.21).

The datasets are shown in Fig. 6.1: the initial mean is chosen to have the same initial measurement value Y_0 for both datasets, and the influence of the initial spread and the noise are visible. Kalman's observability criterion shows that (6.20) is observable, but not (6.21). For (6.21), we have $C^\perp = \text{span}\{(-2, 1)\}$ and $x_b = x_a - (-2, 1)$, so that x_b is indistinguishable from x_a as per Remark 15. However, the presence of noise makes this difficult to see from output trajectories: from a visual inspection of Fig. 6.1, it is not clear that the trajectories of Fig. 6.1b are from the same distributions while those of Fig. 6.1a are not. In contrast, the test of Prop. 2 *is able to find this*: it triggers in the first case but not in the second (with $\sigma = 30$ resp. 80), which is consistent with the analytical results. This also illustrates that statistical testing can assess absolute distinguishability from noisy output data.

Recovering the classes of indistinguishability

We now show that the test can recover a state's whole empirical class of indistinguishability. According to Th. 6, this class should be the same for a stochastic system and the corresponding nominal system. We verify this numerically on the following continuous-time LTI system:

$$\begin{aligned} \mathrm{d}X_t &= AX_t\mathrm{d}t + A_0 \sin(\omega t)\mathrm{d}t + \Sigma \mathrm{d}W_t, \\ Y_t &= CX_t + \epsilon_t, \end{aligned} \tag{6.22}$$

where

$$A = \begin{pmatrix} -2 & -1 \\ -1 & -2 \end{pmatrix}, \qquad C = \begin{pmatrix} -1 & 1 \end{pmatrix}^\top, \tag{6.23}$$

$\mathrm{d}W_t$ is an independent, two-dimensional Wiener process, $A_0 = (3,3)$, $\omega = 2$, $\Sigma = 0.1\mathbb{I}_2$, and $\epsilon_t \sim \mathcal{N}(0, 0.01)$ is Gaussian measurement noise. The term $A_0 \sin(\omega t)dt + \Sigma dW_t$ is a non-centered process perturbation. Kalman's observability criterion shows that the undisturbed linear system (A, C) is not observable. We have $C^\perp = \mathrm{span}\{(1,1)\}$ and $(CA)^\perp = \mathrm{span}\{(1,1)\}$. Thus, according to Remark 15, the class of indistinguishability of x_a for the nominal system is $x_\mathrm{a} + \mathrm{span}\{(1,1)\}$. We show experimentally that the non-centered noise in (6.22) preserves this class of indistinguishability.

We select a reference point $x_\mathrm{a} = (1.5, 0.5)$ and simulate $n = 30$ output trajectories for 2 s starting from x_a, with time steps of length $\Delta t = 0.01$ s, yielding discrete trajectories of length $T = 200$. These trajectories constitute a dataset of samples from $\mathbb{P}^T_{x_\mathrm{a}}$. We emphasize that the process noise has nonzero mean over the simulation time.

We then build a grid of 50×50 points and simulate $m = 30$ output trajectories starting from each point x_b in that grid. We compute the MMD between the dataset from x_a and the one from each x_b, using a squared exponential kernel with $\sigma = 5$. The results are depicted in Figure 6.2. We observe that the test of Prop. 2 does not trigger along the diagonal $x_\mathrm{a} + C^\perp$ (red dots), and triggers everywhere else, empirically identifying this set as the class of *distributional* indistinguishability of x_a given the output data. This indeed corresponds to the class of *deterministic* indistinguishability of x_a for the unperturbed version of (6.22), as predicted by Theorem 6. We observe that not only does the test recover this class, but the MMD acts as a distance from it: the farther x_b is from $x_\mathrm{a} + C^\perp$, the higher the MMD.

6.4.2 Analyzing observability in the state-space

The proposed approach can also be used to analyze distinguishability quantitatively, directly in the state-space, for arbitrary nonlinear systems. We illustrate this with an undamped, unforced Duffing oscillator, often used in nonlinear observer design [203]:

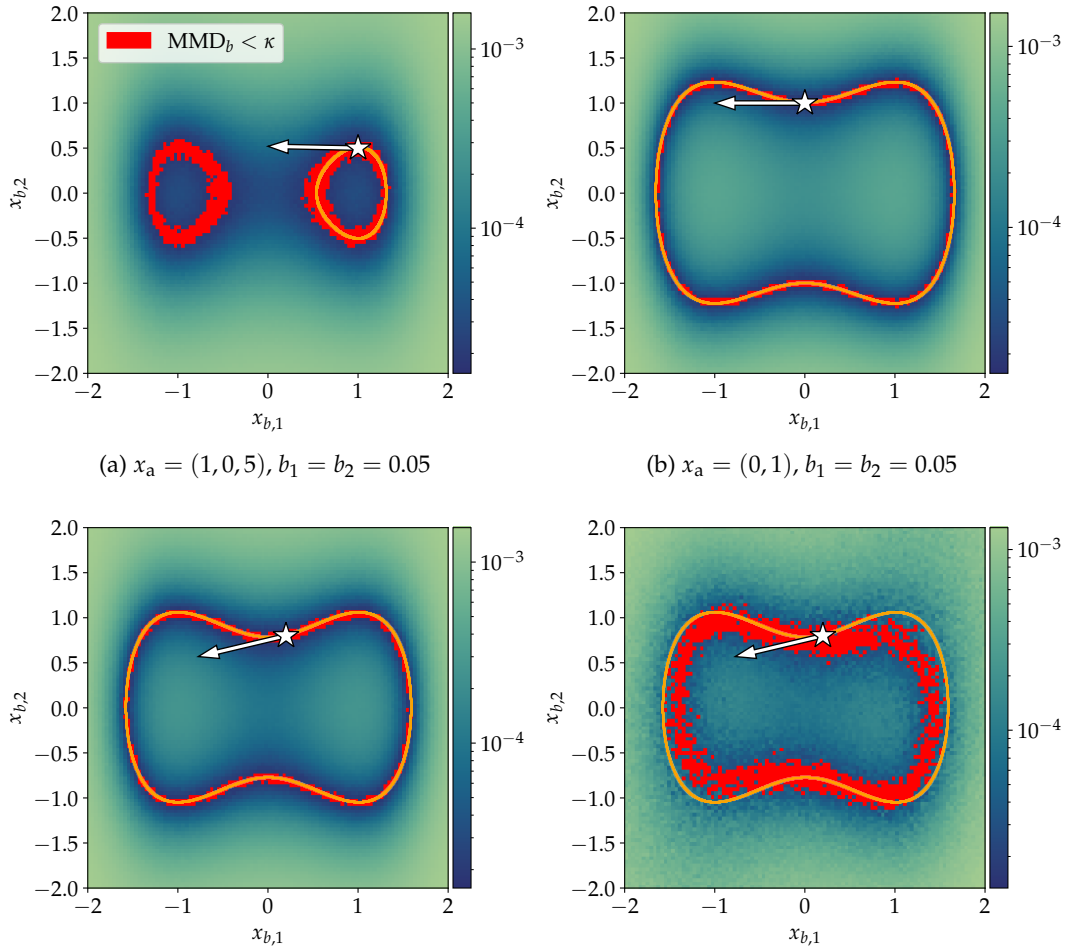
$$\begin{aligned} \mathrm{d}X_1 &= X_2 \mathrm{d}t + b_1 \mathrm{d}W_{1,t}, \\ \mathrm{d}X_2 &= (X_1 - X_1^3)\mathrm{d}t + b_2 \mathrm{d}W_{2,t}, \end{aligned} \tag{6.24}$$

where $\mathrm{d}W_{1,t}$, $\mathrm{d}W_{2,t}$ are independent Wiener processes. The nominal system is Hamiltonian and conserves the quantity

$$h(x) = -\frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 + \frac{1}{4}x_1^4 \tag{6.25}$$

along its trajectories. We consider the output map $Y_t = h(X_t) + \epsilon_t$ with measurement noise $\epsilon \sim \mathcal{N}(0, 0.5)$, and two different settings: low process noise with $b_1 = b_2 = 0.05$ and high process noise with $b_1 = b_2 = 0.5$. Importantly, the trajectories of the stochastic system do *not* conserve h because of process noise.

Given an arbitrary point x_a, we aim to determine its class of indistinguishability from samples of the stochastic system. For this, we generate $n = 50$ output trajectories of 1 s initialized in x_a, and $m = 50$ trajectories in each other point x_b on a 100×100 grid, with $\Delta t = 0.001$ s ($T = 1000$). We then compute the empirical MMD (6.11) between the resulting datasets, with a squared exponential kernel and $\sigma = 1500$.



(a) $x_a = (1, 0, 5)$, $b_1 = b_2 = 0.05$

(b) $x_a = (0, 1)$, $b_1 = b_2 = 0.05$

(c) $x_a = (0.2, 0.8)$, $b_1 = b_2 = 0.05$

(d) $x_a = (0.2, 0.8)$, $b_1 = b_2 = 0.5$

Figure 6.3: MMD over an x_b-grid for the Duffing oscillator (6.24) for different values of x_a (white stars). The trajectory without noise (orange) initialized in x_a is known to be a subset of the class of indistinguishability. The states where the test does not trigger (red points) constitute the empirical class of indistinguishability. The process noise is low in Fig. 6.3a–6.3c, for which only the reference point varies, while the system in Fig. 6.3d is a noisier version of that in Fig. 6.3c. In Fig. 6.3a, the empirical class of indistinguishability contains not only the trajectory starting from x_a, but also its symmetric w.r.t. the origin. In Fig. 6.3d, the empirical class of distinguishability differs from that in Fig. 6.3c due to the significant process noise. In all cases, vectors generating the null space of the nominal system's empirical Gramian (white arrows) are tangent to its class of indistinguishability.

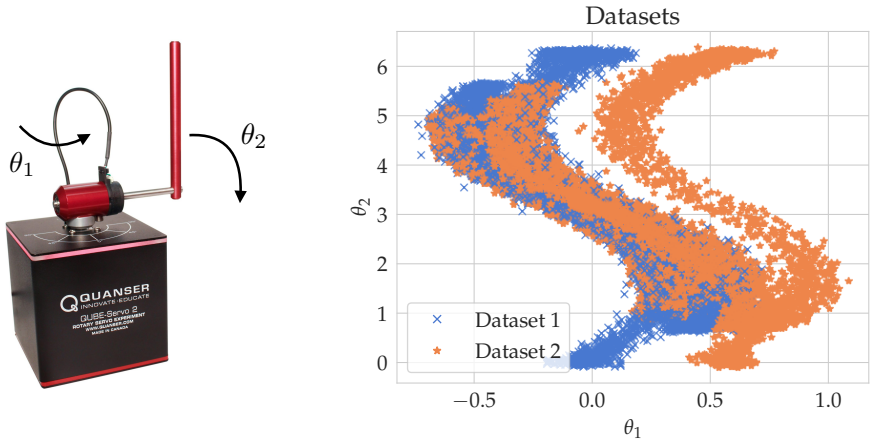


Figure 6.4: Left, the Qube Servo 2 by Quanser [204]. Right, the experimental data collected on this Furuta pendulum. Only the first two states (θ_1, θ_2) are shown. The system is initialized around $x_a = (0, 0, 0, 0)$ for the first dataset, and $x_b = (\pi/4, 0, 0, 0)$ for the second one. Forty trajectories from each dataset are shown (randomly sampled).

Fig. 6.3 shows the resulting MMD maps for two values of x_a and for the different levels of process noise. For low process noise (Fig. 6.3a–6.3c), the MMD increases as the discrepancy in h in the initial states does; the states get more easily distinguishable. Additionally, the empirical class of indistinguishability (red squares) recovers the nominal system's one, which indeed contains the deterministic trajectory starting from x_a (orange curve), where h is constant. For higher levels of process noise (Fig. 6.3d), the empirical class is deformed, illustrating that Corol. 1 does not generalize without further assumptions. This continuous effect of the level of process noise on the classes highlights the interest of a continuous-valued metric of relative distinguishability; states identified as distinguishable by the test may still be *difficult* to distinguish. These plots thus show which regions of the state-space are more or less distinguishable from the chosen x_a, from short and noisy output trajectories.

Remark 16. *For comparison, we also compute the empirical observability Gramian of the nominal system at x_a [192, Eq. 4]. If the null space N of this empirical Gramian is $\{0\}$ in the limit $\epsilon \to 0$, then the system is weakly observable in x_a [192, Thm. 2]. Based on [192, Eq. (14)], we thus interpret N as the "direction of weak unobservability" in x_a. We approximate the assumption $\epsilon \to 0$ by taking $\epsilon = 0.1$ and considering the eigenspace of the smallest eigenvalue of the Gramian, which is several orders of magnitude smaller than the largest one. Fig. 6.3 shows the resulting approximation of N by plotting a generating vector. We find that null space is the tangent space to the class of indistinguishability our test finds for low process noise. In other words, empirical Gramians provide the local directions of indistinguishability, while our test recovers the whole class. Recovering the information on local observability of x_a could also be achieved with our test; e.g., by checking the test outcome in an x_b-spherical shell centered in x_a. Indeed, states very close to x_a will never trigger it due to finite-sample approximations, and should thus be excluded. We leave this question of recovering notions of local observability from the test outcome for future work.*

6.4.3 Analyzing sensor configurations on hardware

When studying an experimental system, one can sometimes choose between possible sensor configurations, whose respective benefits may be uncertain. The system may be observable and allow for building an observer to achieve further downstream tasks in some of these configurations, but not in others. We now demonstrate how the proposed tools can help compare these configurations from measurements. For this, we collect hardware data on a Furuta pendulum: the Qube Servo 2 by Quanser [204], depicted in Fig. 6.4. The pendulum consists of two arms, the first of which is actuated, with respective angular positions $(\theta_1, \theta_2) \in \mathbb{R}^2$ (in radians), where $\theta_1 = 0$ is arbitrary and $\theta_2 = 0$ corresponds to the upright position. The state is then $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$. We collect a set of output trajectories by initializing the system by hand close to $x_\mathrm{a} = (0, 0, 0, 0)$ resp. $x_\mathrm{b} = (\pi/4, 0, 0, 0)$. We measure both angles with encoders and corrupt these measurements with centered Gaussian noise of variance $\sigma_\epsilon^2 = 0.001\,\mathrm{rad}^2$ independently on each dimension, in order to further complicate the experiment. The output trajectories of 9.75 s are sampled with $\Delta t = 0.01\,\mathrm{s}$[3]. A subset of the resulting dataset is plotted in Fig. 6.4.

Choosing a sensor configuration

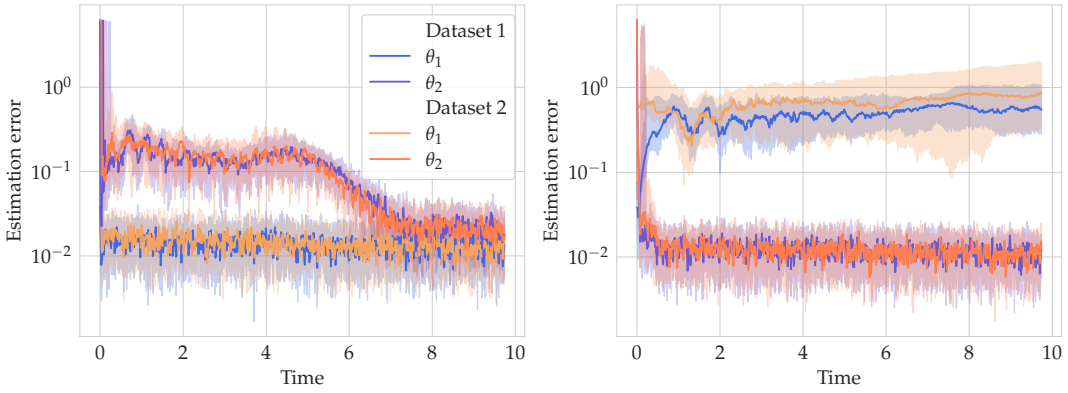
We consider three possible settings: $y = (\theta_1, \theta_2)$, $y = \theta_1$ and $y = \theta_2$. Our goal is to determine from the collected samples which of these configurations enable distinguishing the initial states, a necessary condition to provably reconstruct the whole state of the system.

For this, we run the kernel two-sample test of Prop. 2 ten times in all three settings, and present the results in Table 6.1. The value of σ is chosen as per the proposed meta-heuristic, and given in Table 6.1. The test triggers every time when $y = (\theta_1, \theta_2)$ or $y = \theta_1$, indicating that the initial distributions are distinguishable with these outputs with high confidence. In contrast, it never triggers when $y = \theta_2$; the data does not allow to distinguish between x_a and x_b when only measuring θ_2. This generalizes to the nonlinear system the conclusion obtained by linearizing the dynamics; Kalman's observability criterion states that the model is not locally observable around the origin for $y = \theta_2$, while it is in the other configurations.

Remark 17. *The test evaluates the distinguishability of the initial* distributions μ_a *and* μ_b *whereas, in practice, one is often concerned with the distinguishability of the initial* states x_a *and* x_b*; yet,* μ_a *and* μ_b *are not Diracs. This could lead to a false interpretation of the test result. For example, if* $x_\mathrm{a} = x_\mathrm{b}$ *but* $\mu_\mathrm{a} \neq \mu_\mathrm{b}$ *(e.g., one has much more spread than the other), the output distributions may differ: the test would then trigger, although* x_a *and* x_b *are indistinguishable. To interpret the test result as distinguishability induced by the* dynamics *and not as an artifact introduced by non-ideal initial distributions, the support of the latter should be as narrow as possible around* x_a *and* x_b*. This is theoretically sound due to the continuity of the MMD: if the support of* μ_a *and* μ_b *is sufficiently small then* $\mathrm{MMD}[\mathbb{P}^T_{\mu_\mathrm{a}}, \mathbb{P}^T_{\mu_\mathrm{b}}]$ *is close to* $\mathrm{MMD}[\mathbb{P}^T_{x_\mathrm{a}}, \mathbb{P}^T_{x_\mathrm{b}}]$.

In practice, several methods can ensure a small spread, such as careful reinitializations of the system or discarding trajectories that are too far from the desired initial states. We initialize the system by hand, such that some randomness is always present, but stay in the same experimental conditions for all trajectories. To demonstrate robustness to this issue, we run our experiments multiple times by selecting only $n = m = 40$ trajectories randomly from each initial distribution.

[3] Thanks to Sebastian Giedyk for his help collecting the experimental data.



(a) $y = \theta_1$ (b) $y = \theta_2$

Figure 6.5: Estimation error of the EKF for each choice of measurement, for states (θ_1, θ_2) and on each dataset. For each run, we only consider the mean of the EKF, then plot the median and $\{0.25, 0.75\}$-quantiles over the ten runs, i.e., ten subsets $m = n = 40$ of the experimental datasets. The EKF correctly reconstructs the state in the observable case (left). When $y = \theta_2$ (right), the state θ_1 is not observable, the EKF does not converge correctly and the estimation error remains high.

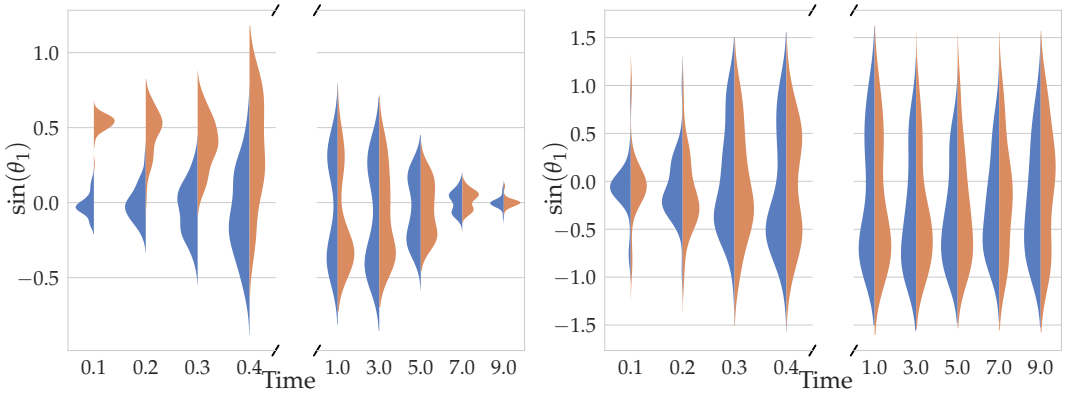
Measurement	$y = (\theta_1, \theta_2)$	$y = \theta_1$	$y = \theta_2$
σ	1300	1300	2500
$\frac{\mathrm{MMD}_b}{\kappa}$	1.40	2.50	0.34
Trigger (% of trials)	100	100	0

Table 6.1: Results for the Furuta pendulum initialized in $x_a = (0, 0, 0, 0)$ and $x_b = (\pi/4, 0, 0, 0)$, averaged over ten tests. Only the first two configurations are identified as distinguishable.

Consequences for observer behavior

Beyond choosing sensor configurations, the information provided by our test can also help in understanding the behavior of observers that fail to converge. An observer initialized in two distributionally indistinguishable initial states takes as input the same output distributions. Therefore, the output of the observer itself has the same distribution; its estimates are the same on the whole class of indistinguishability. In other words, the observer fails to distinguish the states.

We demonstrate this by implementing an extended Kalman filter (EKF) based on a model of the pendulum and taking y as input. We run the EKF on our data set in the two settings $y = \theta_1$ (distinguishable) and $y = \theta_2$ (indistinguishable). The observer converges in the first case, but not in the second one (Fig. 6.5). More interestingly, we estimate the evolution of the distribution of $\sin(\theta_1)$ as estimated by the EKF (Fig. 6.6). In the distinguishable case (Fig. 6.6a), the two output distributions of the EKF differ, whereas they appear to be the same in the indistinguishable case (Fig. 6.6b). This gives a precise meaning to the idea that an observer can estimate the class of indistinguishability of the initial state, which our test can determine.



(a) $y = \theta_1$ (b) $y = \theta_2$

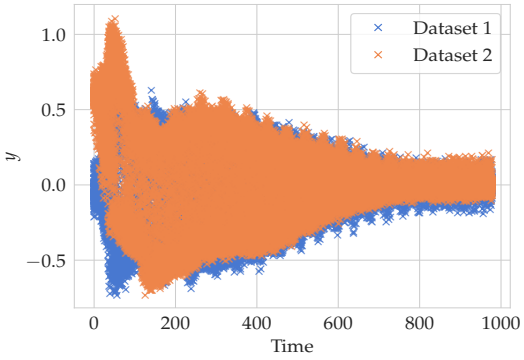
Figure 6.6: Evolution of the distribution of $\sin(\theta_1)$ estimated by the EKF. For each run, we only consider the mean of the EKF, then interpolate the distributions by kernel density estimation over the ten runs. In the distinguishable case (left), the distributions differ during the transient and converge as the measurements do. In the indistinguishable case (right), the distributions remain similar; the output of the observer is statistically the same for both initial states.

6.4.4 Influence of hyperparameters

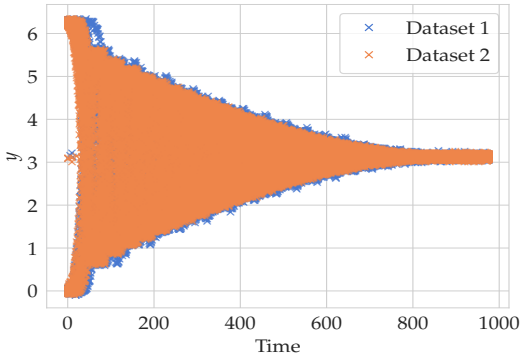
Some hyperparameters play an important role in the observed results. In this section, we discuss the influence of the signal-to-noise ratio in more detail. However, this is not the only relevant parameter. For example, the width σ for the squared exponential kernel is discussed at the beginning of Sec. 6.4. Data normalization, the choice of the grid for the MMD heatmaps and the number of trajectories per point are also crucial. For these, it is worth noting that generating or gathering data is often a costly process: more data will most of the time make the statistical analysis more accurate, but at a high cost. The length T of the measured trajectories also affects the results: as T grows, the states can become more distinguishable, e.g., if the trajectories become different only for T long enough, or less distinguishable, e.g., if the system has a limit cycle and all trajectories become similar for long T. This can lead to different MMD profiles. We do not discuss these choices further, and focus on the role of the signal-to-noise ratio in the following.

We illustrate its influence by modifying the variance σ_ϵ^2 of the Gaussian measurement noise added to the dataset of the Furuta pendulum. The resulting datasets are presented in Fig. 6.7. In Table 6.2, we show the percentage of tests that trigger for these values of σ_ϵ^2 (from ten trials). For $y = \theta_2$, σ_ϵ^2 does not influence the test result: the initial states are indistinguishable even without noise. For $y = \theta_1$, the test remains accurate even with $\sigma_\epsilon^2 = 1$ though the datasets already seem indistinguishable to the human eye. Nevertheless, the performance drops if the noise gets too high, as we can see for $\sigma_\epsilon^2 = 10$. Interestingly, the accuracy decreases more quickly with $y = (\theta_1, \theta_2)$: for $\sigma_\epsilon^2 = 0.1$, the test triggers in less than half of the trials. We interpret this as the distinguishable data, i.e., θ_1, getting diluted into more indistinguishable data by also measuring θ_2. This can also be seen in Table 6.1, where the ratio of the MMD over the test threshold averaged over ten trials is closer to one when measuring both θ_1 and θ_2 than when measuring only θ_1.

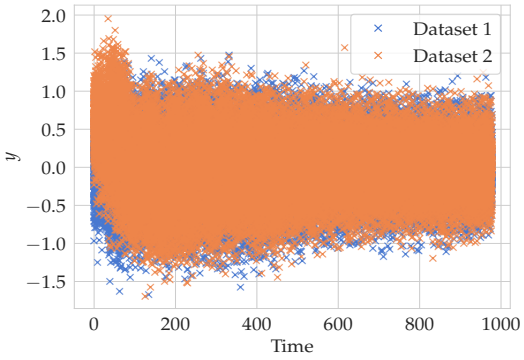
These parameters greatly influence not only the result of the test, and therefore the shape of the empirical classes of indistinguishability, but also the value of the



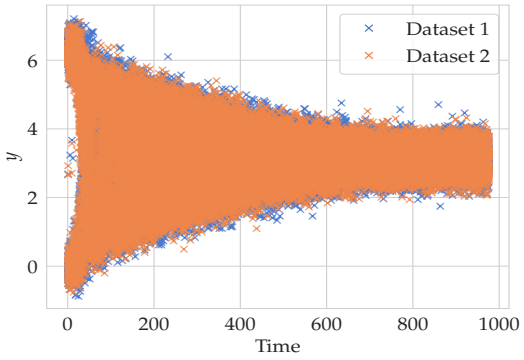
(a) $y = \theta_1$ with $\sigma_\epsilon^2 = 0.001$



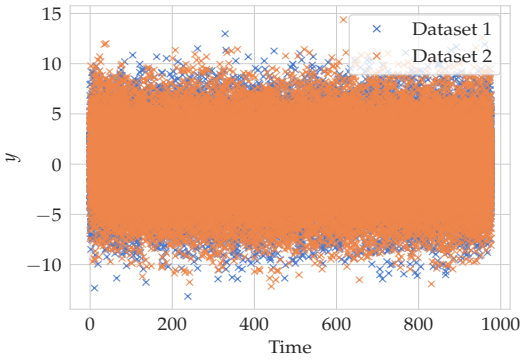
(b) $y = \theta_2$ with $\sigma_\epsilon^2 = 0.001$



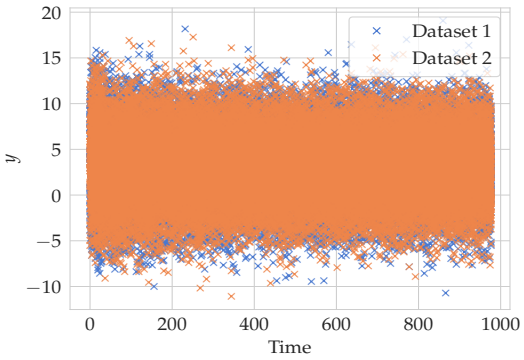
(c) $y = \theta_1$ with $\sigma_\epsilon^2 = 0.1$



(d) $y = \theta_2$ with $\sigma_\epsilon^2 = 0.1$



(e) $y = \theta_1$ with $\sigma_\epsilon^2 = 10$



(f) $y = \theta_2$ with $\sigma_\epsilon^2 = 10$

Figure 6.7: Trajectories of the Furuta pendulum corrupted by different levels of Gaussian measurement noise of variance σ_ϵ^2. After recording, the start of all experiments is realigned and they are interpolated over the same time axis. For some experiments, $\theta_2(0)$ is close to π; this is due to this interpolation between values close to 2π and values close to zero at the start.

Measurement	$y = (\theta_1, \theta_2)$	$y = \theta_1$	$y = \theta_2$
$\sigma_\epsilon^2 = 0.001$	100	100	0
$\sigma_\epsilon^2 = 0.01$	100	100	0
$\sigma_\epsilon^2 = 0.1$	40	100	0
$\sigma_\epsilon^2 = 1$	10	100	0
$\sigma_\epsilon^2 = 10$	0	20	0

Table 6.2: Results for the Furuta pendulum initialized in $x_a = (0, 0, 0, 0)$ and $x_b = (\pi/4, 0, 0, 0)$ for different levels of measurement noise. The percentage from ten trials in which the test triggers is shown. The first two settings are distinguishable, but the third is not.

metric itself, and therefore the MMD heatmaps. However, these heatmaps are less sensitive to changes in the hyperparameters. Indeed, these changes may lead to more or less accurate MMD estimates, but the *relative* values remain similar. Thus, the shape of the heatmap also does, and can be straightforwardly interpreted in terms of low or high *relative* distinguishability w.r.t. the reference point. On the other hand, the statistical test returns a binary answer on the *absolute* distinguishability; higher or lower MMD due to slight variations of these hyperparameters can go over or under the test threshold κ and change this answer.

As an example, we take $x_a = (0, 1)$ and $x_b = (\sqrt{1+\sqrt{3}}, 0)$ for the Duffing oscillator (6.24), which both lead to $h(x) = 0.5$ constant along nominal trajectories. Hence, these two points are indistinguishable for the nominal system. We generate $m = 1000$ trajectories initialized in each of these points. Without any measurement noise, these output trajectories are not quite equal due to the accumulation of numerical errors, and the test identifies them as distinguishable. With little Gaussian measurement noise (variance $\sigma_\epsilon^2 < 0.05$), the trajectories are still identified as distributionally distinguishable. For $\sigma_\epsilon^2 > 0.05$, they become distributionally indistinguishable. However, in all three cases, the MMD between the reference point x_a and the point x_b is low *relatively* to the rest of the grid, as seen in Fig. 6.3b. Thus, the MMD heatmap shows a form of the indistinguishability class of x_a that includes x_b. Therefore, when investigating relative distinguishability in the state-space, which is expected to vary continuously, one should rather focus on the MMD heatmap than on the test results. Conversely, when searching for a binary answer on absolute distinguishability, one can choose a few points that are representative of the system in some sense, and gather data from these points to run the statistical test; this allows for sensor configuration studies as in Sec. 6.4.3.

6.5 Conclusion and outlook

In this chapter, we present a method to assess the observability of nonlinear stochastic systems from data without explicitly relying on a dynamics model, and without assumptions on the noise; this is the first such method to the best of our knowledge.

We start by extending distributional distinguishability to arbitrary nonlinear stochastic systems and investigate its relationship to deterministic observability, showing that both notions are equivalent for certain classes of systems which include linear systems. We then introduce the MMD and a finite-sample estimate thereof to quantify distinguishability of initial states from noisy output trajectories. We leverage the MMD for a statistical test of distinguishability, hereby interpreting its value as

how much data is required to tell initial states apart with confidence. Finally, we illustrate the proposed tools and notions in simulation and on hardware. We reveal experimentally the relationship between noise and distinguishability, emphasizing the interest of a continuous metric. We expect that these tools can become useful for a priori error analysis of observers and thus benefit observer design. For example, the proposed statistical test could be used for experiment design: as shown in Sec. 6.4.3, it can help determine a priori which sensors are necessary to build an observer. Computing MMD heatmaps as in Sec. 6.4.2 could also ease downstream tasks such as tuning an observer: by comparing the estimation error of the observer with the MMD values, a user could know whether the considered states are easily distinguishable, in which case it is worth spending more effort on tuning the observer, or whether this is hardly the case, so that the estimation error can mainly be explained by the lack of observability.

One important assumption we make is the access to independent reinitializations of the system. However, this is data-inefficient (since a whole trajectory only informs on its initial state), and sometimes unrealistic or unwanted (e.g., if the system cannot be reinitialized easily). An interesting future lead is to alleviate this assumption, e.g., by leveraging mixing properties [112].

The theoretical analysis of distributional observability that we propose can also be pushed further, e.g., by characterizing other classes of measurement noise that satisfy Assumption 7, or by finding sufficient conditions on the dynamics such that classes of process noise preserve distinguishability.

After investigating in this chapter how statistical tools can be used to analyze observability properties from output data, we leverage modern machine learning techniques to design observers. In the next chapter, we examine how deep learning can be used to approximate KKL observers and how to tune such numerical observers.

Chapter 7
Towards gain tuning for numerical KKL observers

Résumé Ce chapitre présente une première étape vers le tuning d'observateurs généraux pour les systèmes non linéaires. En s'appuyant sur des résultats récents concernant les observateurs de Kazantzis-Kravaris/Luenberger (KKL), nous proposons un critère empirique pour guider la calibration de l'observateur, en cherchant un compromis entre la performance transitoire et la sensibilité au bruit de mesure. Nous paramétrons la matrice de gain et évaluons ce critère sur une famille d'observateurs pour différentes valeurs du paramètre. Nous utilisons ensuite des réseaux neuronaux pour apprendre la fonction reliant l'état de l'observateur à l'état dans les coordonnées physiques, et présentons une nouvelle méthode pour échantillonner efficacement l'espace d'état pour la régression. Nous illustrons les mérites de cette approche par des simulations numériques.

Abstract This chapter presents a first step towards tuning observers for general nonlinear systems. Relying on recent results around Kazantzis-Kravaris/Luenberger (KKL) observers, we propose an empirical criterion to guide the calibration of the observer, by trading off transient performance and sensitivity to measurement noise. We parametrize the gain matrix and evaluate this criterion over a family of observers for different parameter values. We then use neural networks to learn the mapping between the observer and the physical coordinates, and present a novel method to sample the state-space efficiently for regression. We illustrate the merits of this approach in numerical simulations.

Parts of this chapter are under review under the title *Towards Gain Tuning for Numerical KKL Observers* [52]. This work was conducted in collaboration with Lukas Bahr, who pursued his master's thesis jointly at the Centre Automatique et Systèmes - Mines Paris, PSL and at the Institute for Data Science in Mechanical Engineering, RWTH Aachen University, under the supervision of Mona Buisson-Fenet. He contributed as a co-author of the submission, focusing mostly on implementation and on developing a toolbox for numerical KKL[1].

7.1 Introduction

As discussed in previous chapters, the measurements gathered on physical platforms are often partial and noisy. Estimating the underlying state from these partial observations is thus a critical task in system theory. However, there is no established observer design with convergence guarantees for general nonlinear systems. The Kazantis-Kravaris / Luenberger (KKL) observer could be such a general observer, and has recently gained interest. For such observers to be applicable in practice,

[1]Toolbox available at github.com/Centre-automatique-et-systemes/learn_observe_KKL.git.

many aspects should be investigated, including how to tune their free parameters. Therefore, in this chapter, we propose a numerical method to calibrate KKL observers.

The original design of Luenberger observers for linear systems can be found in [205]. It consists in finding a linear mapping between the system dynamics and a linear filter of the measurement. Under appropriate observability assumptions and filter design, the Sylvester equation satisfied by the mapping has a unique injective solution. Its left-inverse, along with the filter, can be used to compute a convergent state estimate.

This design encompasses important degrees of freedom: the matrices defining the filter or, equivalently, the poles and zeros of the filter transfer function. To study their effect on state estimation performance, one must consider the effect of the mapping, which modifies the response, among others to measurement noise. For autonomous linear systems, the problem of tuning these degrees of freedom is essentially solved by the stationary Kalman filter [206]. Rather than directly assigning closed-loop eigenvalues, one can weigh the relative confidence in the measurement and the dynamic model and find the observer gains that are optimal for the metric defined by these weights.

The extension of these approaches to nonlinear systems is nontrivial. Indeed, there are few generic nonlinear observer designs; a review of these can be found in [42], [43]. Among the most commonly used are the high-gain observer (HGO) [207], [208] and the extended Kalman filter (EKF) [45]. The EKF consists in linearizing the observer dynamics around the current estimate to compute the optimal gain depending on chosen weights, akin to the linear case. There are, however, only local convergence guarantees [209]. Conversely, the HGO relies on a change of variables to bring the system into observable canonical form, and high gains to "dominate" the Lipschitz constant of the nonlinearity. The stability guarantees come at the price of possibly poor transient performance (the so-called "peaking" phenomenon [210]) and high sensitivity to noise. While recent contributions aim at reducing these detrimental features thanks, e.g., to dynamic extensions [120], the question of gain tuning and performance criteria remains open. In particular, in [120], the sensitivity to noise is examined a posteriori through numerous simulations.

In this chapter, we develop a tuning methodology for KKL observers that does not rely on extensive tests, inspired by the Kalman filter or H_∞ control. The KKL design [46], [47] extends the results of [205] to nonlinear systems. It maps the system dynamics to a stable linear filter of the measured output, called the observer dynamics. The existence and injectivity of this mapping are guaranteed by mild observability conditions, which makes this design relatively generic. The contraction properties of the observer dynamics ensure convergence of the state estimates. The main challenge consists in computing said mapping and its left inverse, along with tuning the free parameters of the observer.

In [116], a method is proposed to approximate the mapping by performing nonlinear regression on datasets generated from trajectories of the system and the observer. Given fixed observer parameters, a neural network approximates the considered mapping, which is then used to compute state estimates from observer values. This methodology, which we denote by numerical KKL and illustrate in Fig. 7.1, makes KKL observers applicable even when the transformations cannot be computed analytically. However, the free parameters of the observer still play an essential role and need to be chosen appropriately.

In this chapter, we build on the approach of [116] and propose a first step towards calibration of the observer. Our main contribution is a procedure to select the gain matrix using a tuning criterion that, in some sense, trades off the transient

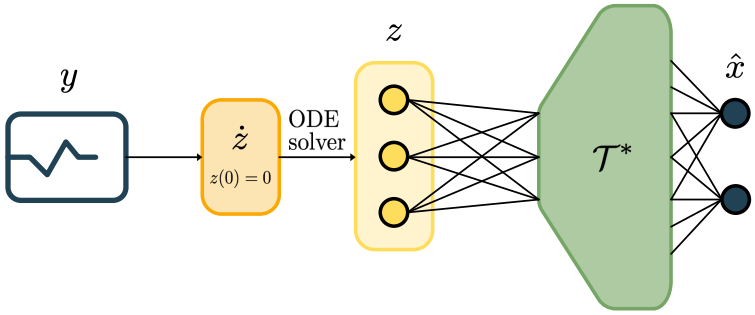


Figure 7.1: Diagram of the learned KKL observer. First, we solve the ordinary differential equation (7.2) for the measurement y generated from the original system (7.1). Then, the estimate $\hat{x}$ is computed as $\mathcal{T}_\theta^*(z)$, where $\mathcal{T}_\theta^*$ approximates $\mathcal{T}^*$.

performance against the sensitivity to measurement noise. We start by setting this matrix based on a pre-defined filter, parametrized by its cut-off frequency ω_c. We then approximate the KKL mapping for different values of this parameter ω_c using neural networks, either independently or by learning the mapping as a function of ω_c. This approximation is enabled by appropriately sampling the state-space, improved upon [116]. Computing the proposed criterion for all values of the parametrized gain matrix leads to an optimal calibration for the observer, in the sense of the proposed empirical criterion. Numerical simulations illustrate the approach.

This chapter is organized as follows. In Sec. 7.1.1, we briefly recall the main equations of KKL observers. In Sec. 7.2, we propose an empirical gain tuning criterion, then detail our numerical approach for state-space sampling, observer parametrization and nonlinear regression in Sec. 7.3. We illustrate the merits of the approach through numerical simulations in Sec. 7.4. Then, we investigate another point of view using autoencoders in Sec. 7.5, before concluding in Sec. 7.6

7.1.1 Reminder on KKL observers

Recall the preliminaries on KKL observers presented in Sec. 3.4. We consider the autonomous nonlinear dynamical system

$$\begin{aligned} \dot{x} &= f(x) \\ y &= h(x) \end{aligned} \tag{7.1}$$

where $x \in \mathbb{R}^{d_x}$ is the state, $y \in \mathbb{R}^{d_y}$ is the measured output, f is a continuously differentiable function (C^1) and h is a continuous function. Assume there exists a compact set $\mathcal{X}$ such that for any solution of interest x to (7.1), $x(t) \in \mathcal{X}$ for all $t \geq 0$. Under the assumptions and results presented in Sec. 3.4, there exists a continuous injective mapping $\mathcal{T} : \mathbb{R}^{d_x} \to \mathbb{C}^{d_z}$ and its continuous left inverse $\mathcal{T}^* : \mathbb{C}^{d_z} \to \mathbb{R}^{d_x}$ such that the trajectories of (7.1) remaining in $\mathcal{X}$ and any trajectory of

$$\dot{z} = Dz + Fy \tag{7.2}$$

satisfy

$$\lim_{t \to +\infty} |\mathcal{T}^*(z(t)) - x(t)| = 0. \tag{7.3}$$

Equivalently, for all trajectories of (7.1) remaining in $\mathcal{X}$, $\mathcal{T}$ is an injective solution of the following partial differential equation (PDE):

$$\frac{\partial \mathcal{T}}{\partial x}(x)f(x) = D\mathcal{T}(x) + Fh(x). \tag{7.4}$$

Thus, implementing a KKL observer involves the following steps:

1. Choose matrices D and F
2. Compute the corresponding transformation $\mathcal{T}^*$
3. Simulate (7.2) from an arbitrary $z(0)$ and compute the estimate $\hat{x}(t) = \mathcal{T}^*(z(t))$.

In [116], a method to complete step 2 by performing nonlinear regression on trajectories of (7.1) and (7.2) is proposed. In this chapter, we build on this supervised approach denoted by numerical KKL and illustrated in Fig. 7.1. We propose to assist the user in completing step 1 by defining an empirical performance criterion to optimize.

7.2 A gain tuning criterion

Consider the dynamical system (7.1) and associated observer dynamics (7.2). Denote x, z their solutions starting resp. at $x(0)$ and $\mathcal{T}(x(0))$. Assume now that the measurement y is corrupted by an unknown noise vector $\epsilon \in \mathbb{R}^{d_y}$, so that $y(t) = h(x(t)) + \epsilon(t)$. Denote $\hat{z}$ the corresponding solution of (7.2) starting at an arbitrary initial condition z_0, and $\tilde{z} = \hat{z} - z$ the estimation error due to both the initial error and the measurement noise. In general, we aim to choose D such that the overall error on the estimated state $\hat{x}$ is minimized, where $\hat{x} = \mathcal{T}^*(\hat{z})$, similarly to [211]. The following result provides a criterion for tuning D, which we then apply to the approximated transformation.

Proposition 3. *Suppose the assumptions and results of Sec. 3.4 hold. Further, assume that $\mathcal{T}^*$ is Lipschitz continuous of constant L. Then, we have*

$$|\hat{x} - x|_{L^2} \leq L\Big(|G_\epsilon|_\infty |\epsilon|_{L^2} + |G_z|_{H^2} |\tilde{z}(0)|\Big) \tag{7.5}$$

where $|\cdot|$ is the Euclidean norm, $|\cdot|_{L^2}$ and $|\cdot|_{H^2}$ are the L^2 resp. H^2 norms, and the H_∞ norm is defined as

$$|G|_\infty = \sup_\omega |G(j\omega)| \tag{7.6}$$

with $G_\epsilon(s) = (s\mathbb{I}_{d_z} - D)^{-1}F$ the transfer function from ϵ to $\tilde{z}$, and $G_z(s) = (s\mathbb{I}_{d_z} - D)^{-1}$ from $\tilde{z}(0)$ to $\tilde{z}$.

Proof. By Lipschitz continuity of $\mathcal{T}^*$, we have

$$\begin{aligned} |\hat{x} - x|_{L^2}^2 &= \int_0^\infty |\mathcal{T}^*(\hat{z}(t)) - \mathcal{T}^*(z(t))|^2 dt \\ &\leq L^2 |\tilde{z}|_{L^2}^2. \end{aligned} \tag{7.7}$$

The Laplace transform applied to the dynamics of $\tilde{z}$ yields

$$\begin{aligned} \underline{\tilde{z}}(s) &= (s\mathbb{I}_{d_z} - D)^{-1}F\underline{\epsilon}(s) + (s\mathbb{I}_{d_z} - D)^{-1}\tilde{z}(0) \\ &= G_\epsilon(s)\underline{\epsilon}(s) + G_z(s)\tilde{z}(0), \end{aligned} \tag{7.8}$$

where we denote the Laplace transform of a signal $f(t)$ by $\underline{f}(s)$. Applying standard results on signal norms for linear systems [212] yields

$$|\tilde{z}|_{L^2} = |\underline{\tilde{z}}|_{L^2} \leq |G_\epsilon|_\infty |\epsilon|_{L^2} + |G_z|_{H^2} |\tilde{z}(0)|. \tag{7.9}$$

Replacing (7.9) in (7.7) concludes the proof. □

Proposition 3 exhibits a standard trade-off in linear system theory, between sensitivity to noise through the term in $|\epsilon|_{L^2}$ and convergence speed through the term in $|\tilde{z}(0)|$. In this chapter, we propose a heuristic that guides the choice of D such that the error on the estimate $\hat{x}$ is minimized.

Remark 18. *Proposition 3 relies on the assumption that $\mathcal{T}^*$ is Lipschitz continuous. This is not true in general; however, we approximate $\mathcal{T}^*$ with the neural network model $\mathcal{T}^*_\theta$, which is Lipschitz if its activation function is and if its weights are bounded [213]. Its Lipschitz constant can be approximated empirically, for example by computing its maximum over a regular grid of n samples z_j. However, the maximum value is subject to outliers and tends to vary strongly between models.*

In light of this remark, we propose the following empirical criterion

$$\begin{aligned} \alpha(D) &:= |J|\left(|G_\epsilon|_\infty + |G_z|_{H^2}\right) \\ J &:= \left(\left|\frac{\partial \mathcal{T}^*}{\partial z}(z_j)\right|\right)_{j\in\{1,\cdots,n\}} \end{aligned} \tag{7.10}$$

where we consider the Euclidean norm of J rather than its infinity norm. This is an approximate bound for the constants on the right-hand side of (7.5). This heuristic trades off the transient performance through $|G_z|_{H^2}$ and the noise sensitivity through $|G_\epsilon|_\infty$ and $|J|$. In our experiments, we consider a family of matrices D indexed by a scalar parameter ω_c. We compute α for different $D(\omega_c)$ and pick the value of ω_c that minimizes it.

Remark 19. *The bound (7.7) is conservative, and the choice of the L^2 norm is somewhat arbitrary. In practice, one could consider a variety of criteria by weighting different norms of $\frac{\partial \mathcal{T}^*}{\partial z}$, G_ϵ, and G_z. For example, in the linear case where $\mathcal{T}$, $\mathcal{T}^*$ are matrices, we have*

$$\underline{\hat{x}}(s) - \underline{x}(s) = (s\mathbb{I}_{d_z} - \mathcal{T}^* D \mathcal{T})^{-1} \mathcal{T}^* F \underline{\epsilon}(s) + (s\mathbb{I}_{d_z} - \mathcal{T}^* D \mathcal{T})^{-1} (\hat{x}(0) - x(0)). \tag{7.11}$$

Another criterion could be the H_∞ norm of an analogy of this transfer function (7.11) for the nonlinear case using the empirical gradients. Previous works investigate the sensitivity of linear Luenberger observers to measurement noise [214], [215] or even propose iterative schemes to adapt the gain based on the output error for nonlinear systems [216]. However, there seems to be little research on principled tuning guidelines even for linear Luenberger observers.

Remark 20. *Note also that there are more advanced methods for estimating the Lipschitz constant of $\mathcal{T}^*_\theta$ [213]; we focus on the simpler criterion (7.10), which is enough to exhibit some of the trade-offs faced when tuning D.*

In the next section, we present a method to improve the regression process by carefully generating the dataset and propose a possible parameterization of D, before illustrating the merits of the criterion in Sec. 7.4.

7.3 Numerical methods

As in [116], we approximate the transformation $\mathcal{T}^*$ by a neural network[2] of weights θ. The resulting observer is illustrated in Fig. 7.1: we feed the measurement y into the observer dynamics (7.2), then apply the neural network model $\mathcal{T}_\theta^*$. To train $\mathcal{T}_\theta^*$, i.e. perform nonlinear regression, a dataset of N pairs (x_i, z_i), $i \in \{1, \dots, N\}$ needs to be generated from trajectories of (7.1), (7.2). The construction of this dataset represents an important challenge, as the observer state z converges towards $\mathcal{T}(x)$ only after a transient period whose length depends on D. This transient is not suitable for gathering data to learn the transformation, since we do not have $x \simeq \mathcal{T}^*(z)$ during the transient. However, for autonomous nonlinear systems, the trajectories tend to converge towards the ω–limit sets [217] of the dynamics, so that the points (x_i, z_i) after the transient tend to be close to these ω–limit sets, leading to an uninformative dataset. For example, if the system has a limit cycle, then even points selected uniformly over $\mathcal{X}$ will be close to this limit cycle after the transient of the observer, so that not all of the state-space is sampled. We solve this problem in Sec. 7.3.1.

Further, to calibrate the observer using the gain tuning criterion (7.10), the gain matrix D needs to be parametrized by a scalar ω_c. Then, one can either learn a model $\mathcal{T}_\theta^*$ for each value of ω_c independently, or learn the transformation as a function of ω_c. This yields a more difficult regression problem but avoids learning a new transformation each time the pair (D, F) is changed. This is discussed in Sec. 7.3.3.

7.3.1 Backward-forward sampling

The choice of (x_i, z_i) pairs is critical to numerically approximate $\mathcal{T}^*$. In [116], inspired by [218], the authors propose to first generate an arbitrary grid of initial conditions $(x(0), z(0))$ using standard statistical methods such as Latin hypercube sampling (LHS). Then, relying on the observer's stability, meaning that it forgets its arbitrary initial condition $z(0)$ after some time, the dynamics $x(t)$ and $z(t)$ are simulated forward in time for t_c, where t_c is chosen large enough such that $z(t_c)$ is "close" to its steady-state. Finally, the beginning of the solutions $(x(t), z(t))$ for $t < t_c$ is removed from the dataset.

Unfortunately, this approach lets the dynamics dictate the position of the (x_i, z_i) pairs: for large values of t_c, they are bound to be located close to the ω–limit sets of the system [217]. However, it is desirable to have training samples all over the state-space, especially in regions where the function $\mathcal{T}^*$ is less smooth and therefore more difficult to approximate.

We propose the following methodology to generate an arbitrary dataset of (x_i, z_i) pairs.

1. Choose N initial conditions $x_i(0) \in \mathcal{X}$, $i \in \{1, \dots, N\}$ using a uniform grid, LHS, or any other method.
2. Simulate the system $\dot{x} = f(x)$ from $x_i(0)$ *backward in time* for t_c seconds.
3. If the system diverges in backward finite time, then f should be saturated smoothly outside of $\mathcal{X}$ as suggested in [47], [114]. An example of saturation is provided in Sec. 7.4.3.

[2] Note that $\mathcal{T}$ can also be approximated using the same methodology, but is not necessary for state estimation. Hence, we focus on $\mathcal{T}^*$.

4. Simulate both systems $\dot{x} = f(x)$ and $\dot{z} = Dz + Fy$ with $y = h(x)$, starting from $x_i(-t_c)$ obtained previously and $z_i(-t_c) = z_0$, where z_0 is an arbitrary initial condition, for t_c seconds forward in time.

5. Set the training dataset to $(x_i, z_i) = (x_i(0), z_i(0))$ as obtained from backward-forward simulation.

With this approach, the user can set the training points x_i a priori and obtain the corresponding z_i without the system dynamics modifying the desired state-space grid.

7.3.2 Trajectory-based sampling

Due to the curse of dimensionality, a large amount of data is necessary to learn $\mathcal{T}^*$ with $d_x \geq 4$. To limit the computations, one can generate data along realistic trajectories of the system. For this, we select a fixed number of initial conditions $x_i(0)$ (using LHS or any other sampling scheme), use backward-forward sampling to obtain the corresponding $z_i(0)$ values, then run a joint simulation of both the x and z trajectories for a fixed time. Sampling along these trajectories yields a dataset of (x_i, z_i) values. This is to be contrasted with backward-forward sampling: if the system has a limit cycle, many of the samples will be close to it. However, an appropriate choice of the initial states and trajectory length can counterbalance this effect. Overall, this hybrid sampling scheme combining backward-forward sampling and samples along trajectories helps build an informative dataset efficiently, by generating more data in regions where the system goes more often, instead of densely sampling the whole state-space.

7.3.3 Parametrization

To evaluate the proposed gain tuning criterion, we parametrize D by a scalar ω_c. Several parametrizations can be considered, for example choosing D as a given diagonal matrix multiplied by a factor. In this chapter, we propose to use a d_z-order Bessel filter with cut-off frequency $2\pi\omega_c$, while $F = \mathbf{1}_{d_z \times d_y}$ is fixed to guarantee the controllability of (D, F). We choose D by setting its eigenvalues to be the filter's poles. For any set of poles $(p_1, \ldots, p_n)$ where p poles are real and m poles are complex conjugates such that $n = p + 2m$, we choose D as the following block-diagonal matrix:

$$D = \begin{pmatrix} D_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_{p+m} \end{pmatrix}, D_i = \begin{cases} p_i & \text{if } p_i \text{ is real} \\ \begin{pmatrix} \Re p_i & \Im p_i \\ -\Im p_i & \Re p_i \end{pmatrix} & \text{otherwise} \end{cases} \tag{7.12}$$

The choice of parametrization influences the performance of the obtained models. In this chapter, we focus on this Bessel parametrization because it provides a certain physical interpretation – the behavior of Bessel filters is well-known in linear filtering and characterized by the cut-off frequency – and because it shows good performance in our experiments. We have also considered other possibilities such as Butterworth filters, which lead to similar performance, or using a diagonal or identity matrix multiplied by a factor, so that this factor is the parameter to be optimized. However, it is still unclear how to pick a suitable form for D for a given system. Analyzing these different possibilities further could be an interesting topic for future work.

We can then compute the gain tuning criterion (7.10) for different values of ω_c. For this, we can:

(i) learn a model $\mathcal{T}_\theta^*$ for each value of interest;

(ii) learn a family of transformations $\mathcal{T}^*$ as functions of ω_c: the transformation to approximate is then $\mathcal{T}_\theta^*(z, w_c)$.

Option (i) requires training several neural networks independently for each value of D, which can be tedious. Also, if the observer needs to be fine-tuned, a new model will be required. Option (ii) yields a more challenging regression problem, hence training will require more data and a careful design, but also yields a single model for all values of D. The user can then choose an acceptable value of D for the use case at hand and directly use the previous model. Alternatively, they can train again for this specific value of D to obtain a more accurate approximation for this particular choice. This approach can be advantageous for low-dimensional problems or when the observer will be needed in different experimental conditions without re-training.

In the next section, we illustrate the relevance of criterion (7.10) for choosing D in numerical simulations, using the proposed sampling scheme and parametrization.

7.4 Results

We now evaluate the proposed approach on simulations of three nonlinear systems[3]. We demonstrate that D can be tuned a posteriori by optimizing a metric such as (7.10), and show that it is a relevant criterion for choosing D so as to limit the noise sensitivity of the state estimate. We learn the observer as a function of ω_c for the first two systems (option (i)), and independently for different values of ω_c for the third system (option (ii)). Note that the model can eventually be trained again after selecting ω_c to reach higher accuracy.[4]

To simulate the first two systems, we use an explicit Runge-Kutta method of order 4 (RK4) with a fixed time step of 0.001 s; an explicit Runge-Kutta method of order 5(4) with adaptive step size (RK4/5) [1] is used for the third system. This seems accurate enough for backward-forward sampling in our examples, but future users should make sure the chosen solver ensures low numerical error to simulate f backward then forward in time for t_c and obtain the corresponding value of z.

7.4.1 Reverse Duffing oscillator

The reverse Duffing oscillator

$$\begin{cases} \dot{x}_1 = x_2^3 \\ \dot{x}_2 = -x_1 \end{cases} \qquad y = x_1 \tag{7.13}$$

is a nonlinear system whose solutions evolve on invariant compact sets. We choose a set of hundred even-spaced values for ω_{c_i} in $[0.03, 1]$. Then, LHS is used to select $N = 5,000$ samples $x_i \in [-1, 1]^2$ for each value of ω_c. The corresponding z_i samples are computed using backward-forward sampling as described in Sec. 7.3.1. The training data is normalized to ease the optimization process. For each given ω_c, D is computed following (7.12), while $F = \begin{pmatrix} 1, 1, 1 \end{pmatrix}$. The time t_c after which we consider that the observer has converged is set to $\frac{10}{\lambda_{min}(D)}$, where $\lambda_{min}(D)$ is the minimum absolute value of the real part of the eigenvalues of D, such that it is different for

[3]Note that for our empirical criterion (7.10) to be meaningful, the variables should be normalized [219]. In these academic examples, the variables can be considered scaled.

[4]Code to reproduce the results is available at github.com/monabf/learn_observe_KKL.git.

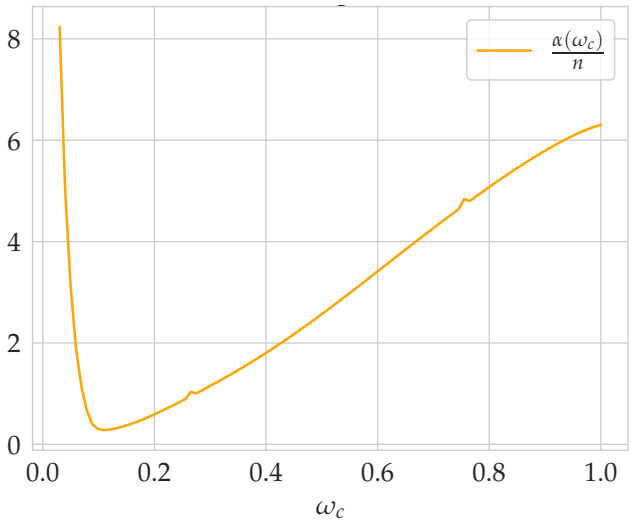


Figure 7.2: Proposed gain tuning criterion (7.10) for the reverse Duffing oscillator, divided by $n = 10,000$ points used to compute $\frac{\partial \mathcal{T}_\theta^*}{\partial z}(z_j)$. The infinity and H_2 norms are high for low values of ω_c, while the gradient of the approximate transformation is high for high values. Choosing $\omega_c = 0.15$ appears to be optimal w.r.t. this metric.

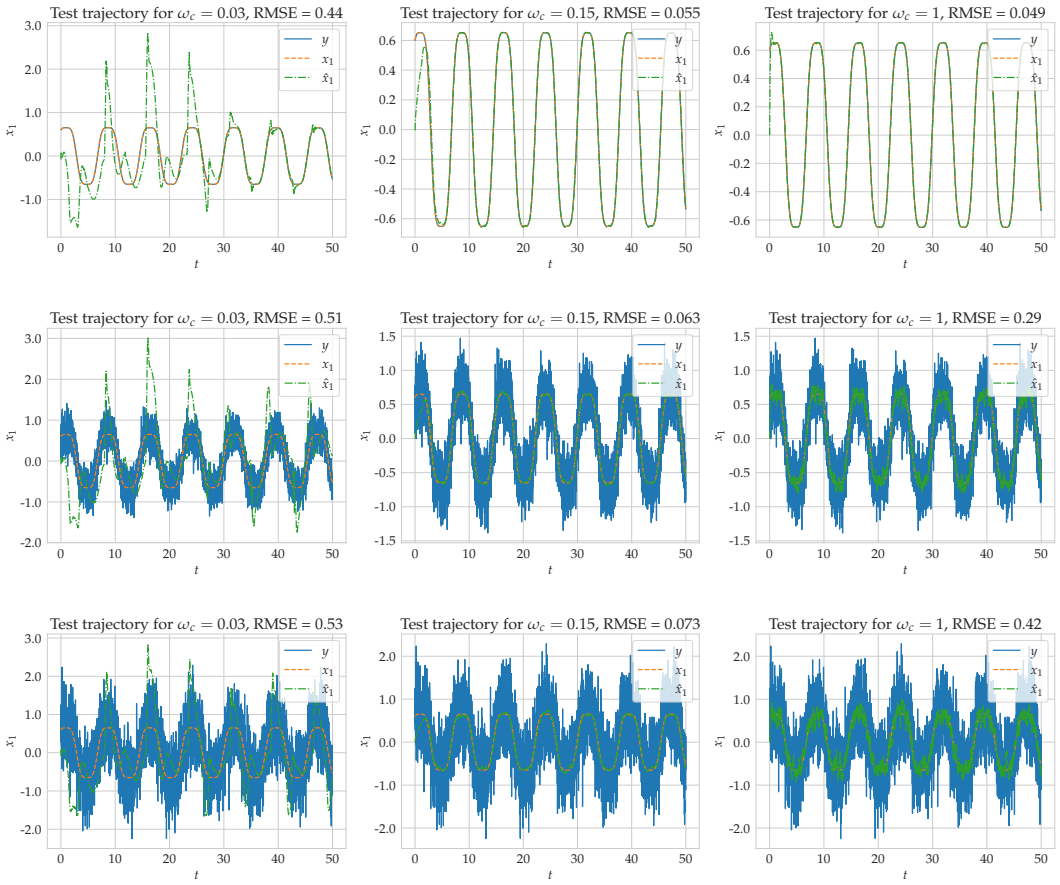
each value of ω_c. The neural networks are multi-layer perceptrons with five hidden layers of size 50 and *SiLU* activation, which is Lipschitz continuous and shows good performance. We train $\mathcal{T}_\theta^*$ by minimizing

$$L_s(\theta) = \frac{1}{2} \sum_{x_i, z_i, \omega_{c_i}} |x_i - \mathcal{T}_\theta^*(z_i, \omega_{c_i})|^2. \tag{7.14}$$

We approximate $\mathcal{T}^*(z, \omega_c)$ over the training data as a function of ω_c, then compute the criterion (7.10) for each value of ω_c over a uniform grid of $n = 10,000$ test points z_j, also obtained with backward-forward sampling from a uniform grid in x. The empirical criterion is shown in Fig. 7.2.

The choice of ω_c greatly influences the performance of the learned observer, as seen in Fig. 7.3. In our simulations, lower values of ω_c lead to a long convergence time and large overshoot, which corresponds to high values of $|G_z|_{H^2}$. However, low ω_c also yields a high signal-to-noise ratio in z, such that the observer is relatively robust to measurement noise. This is illustrated in the left column of Fig. 7.3. On the other hand, high values of ω_c lead to a high gradient of $\mathcal{T}_\theta^*$: the approximate transformation is not smooth and therefore very sensitive to changes in z, hence to measurement noise. The signal-to-noise ratio in z is also low due to the fast eigenvalues of D. This is depicted at the bottom right of Fig. 7.3. In the central column of Fig. 7.3, we select $\omega_c = 0.15$ the optimal value according to criterion (7.10). This setting yields an acceptable trade-off between these different aspects: both transient overshoot and noise sensitivity remain limited. Hence, the proposed gain tuning criterion leads to satisfying performance for this use case.

Further, the transformation seems more difficult to learn for low values of ω_c and less able to extrapolate to unseen values, as seen in Fig. 7.4. A possible explanation is that the existence and injectivity of $\mathcal{T}$ solution to (7.4) is only guaranteed for eigenvalues of D with large enough real parts in absolute value [42], [47], such that low values of ω_c can violate the existence guarantee for $\mathcal{T}^*$. We observe that higher



(a) $\omega_c = 0.03$ (b) $\omega_c = 0.15$ (c) $\omega_c = 1$

Figure 7.3: Estimated trajectories of the reverse Duffing oscillator for $x(0) = (0.6, 0.6)$, without measurement noise at the top, with Gaussian measurement noise $\mathcal{N}(0, 0.25)$ in the middle and $\mathcal{N}(0, 0.5)$ at the bottom. For each setting, we compute the root mean squared error (RMSE) over the whole trajectory. For low ω_c (left), we observe long transients. For high ω_c (right), the estimate is sensitive to high-frequency noise. For $\omega_c = 0.15$ (middle), it is accurate and relatively robust to measurement noise.

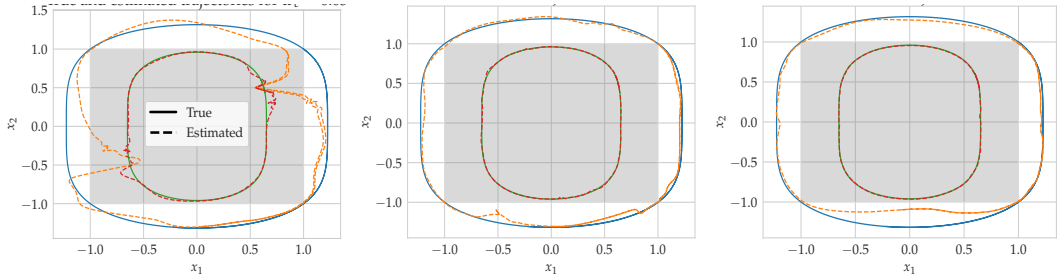
values lead to a better data fit, and better generalization capabilities, even when learning a separate model for each value of ω_c.

7.4.2 Van der Pol oscillator

The autonomous Van der Pol oscillator

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = (1 - x_1^2)x_2 - x_1 \end{cases} \qquad y = x_1 \tag{7.15}$$

admits a unique, globally attractive limit cycle. However, its trajectories diverge in finite backward time, rendering backward-forward sampling numerically intractable.



(a) $\omega_c = 0.03$ (b) $\omega_c = 0.15$ (c) $\omega_c = 1$

Figure 7.4: Two test trajectories of the reverse Duffing oscillator with $x(0) = (0.6, 0.6)$ resp. $x(0) = (1.5, 1.5)$. The first one is inside the training domain $\mathcal{X}$ colored in grey, the second one is outside. We initiate the observer at $z(0) = \mathcal{T}_\theta(x(0))$ to avoid the transient for plotting. With low ω_c, the transformation $\mathcal{T}^*$ is more difficult to learn and less able to extrapolate to unseen points.

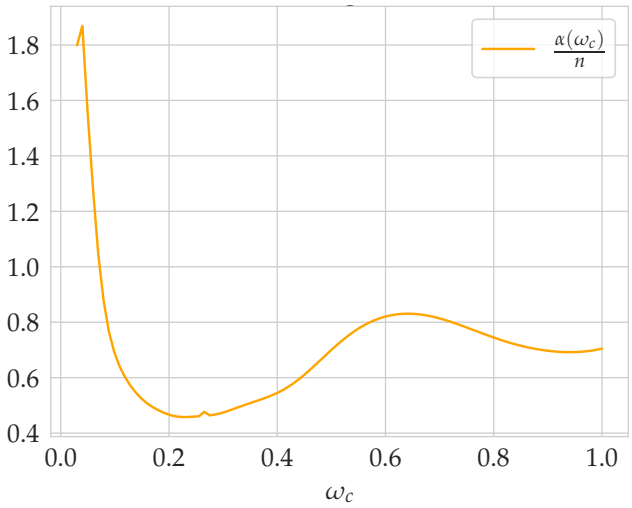


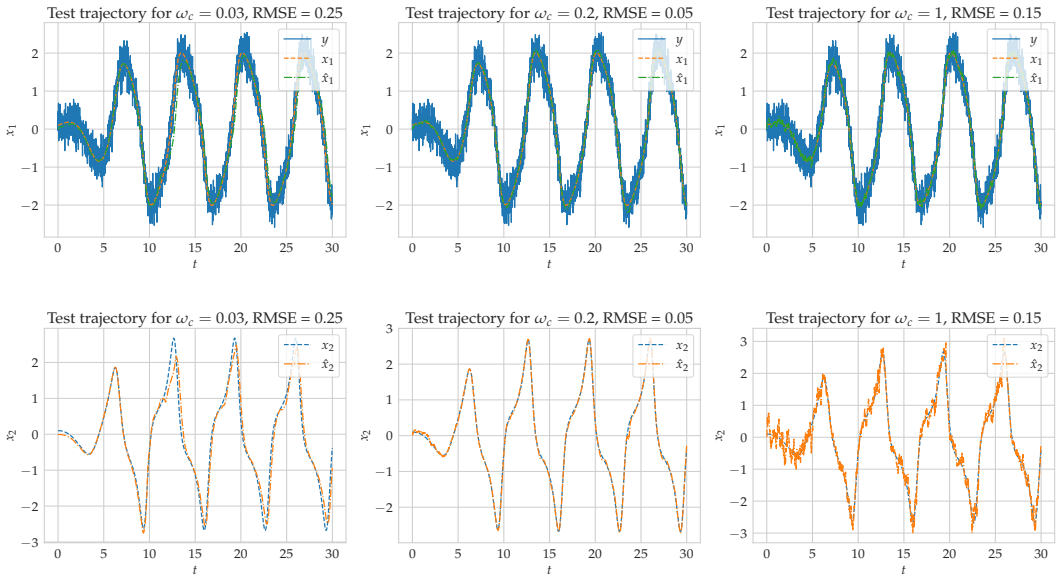
Figure 7.5: Proposed gain tuning criterion (7.10) for the saturated Van der Pol example, divided by $n = 10,000$. Choosing $\omega_c = 0.2$ appears to be optimal w.r.t. this metric.

As suggested in [47], [114], we rather consider the modified system

$$\begin{aligned} \dot{x} &= f(x)g(x), \\ g(x) &= \begin{cases} 1 & \text{if } |x| \leq r \\ 0 & \text{if } |x| \geq r + d \\ p(|x| - r) & \text{otherwise} \end{cases} \end{aligned} \tag{7.16}$$

where f is the dynamics function (7.15) and g is a Lipschitz saturation function. The compact of interest in which we learn the observer is $\mathcal{X} = [-2.7, 2.7]^2$, and we set $r = 3$ and $d = 7$ such that the set outside of which we saturate is $\{x \in \mathbb{R}^2, |x| \leq 3\}$ and the set outside of which the saturated dynamics are zero is $\{x \in \mathbb{R}^2, |x| \leq 10\}$. The function $p(\cdot)$ is chosen as a polynomial of order three such that $p(0) = 1$, $p'(0) = 0$, $p(d) = 0$ and $p'(d) = 0$, which guarantees that g be C^1.

This modified system has the same trajectories as (7.15) inside $\mathcal{X}$ but does not



(a) $\omega_c = 0.03$ (b) $\omega_c = 0.2$ (c) $\omega_c = 1$

Figure 7.6: Estimated trajectories (x_1 above, x_2 below) of the Van der Pol oscillator for $x(0) = (0.1, 0.1)$, with Gaussian measurement noise $\mathcal{N}(0, 0.25)$. For low ω_c (left), we observe long transients and an inaccurate model at the turning points of the limit cycle. For high ω_c (right), the estimate is sensitive to high-frequency noise. Choosing $\omega_c = 0.2$ (middle) yields a compromise between the accuracy of the learned transformation and its sensitivity to measurement noise.

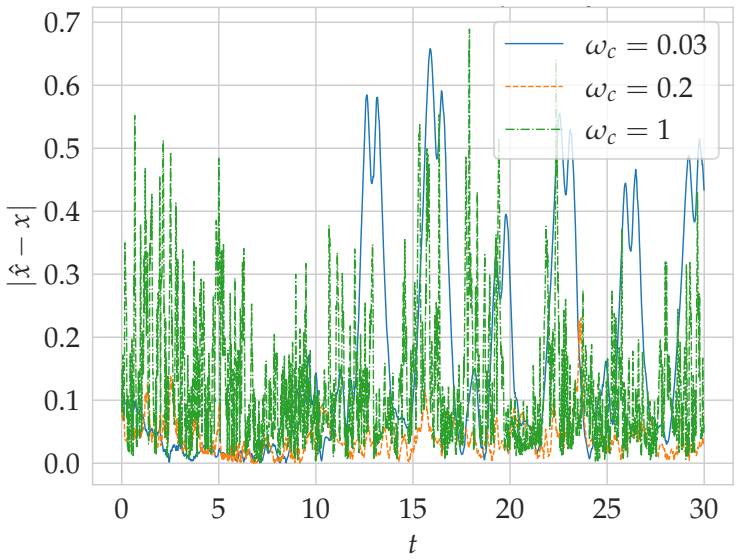


Figure 7.7: Estimation error $|\hat{x} - x|$ for a test trajectory of the Van der Pol oscillator starting at $x(0) = (0.1, 0.1)$, with Gaussian noise $\mathcal{N}(0, 0.25)$ on the measurement.

diverge in backward time from any initial condition in $\mathcal{X}$. As investigated in [114], the existence and injectivity of a suitable transformation $\mathcal{T}$ is guaranteed on $\mathcal{X}$, however this transformation is not unique. In particular, it depends on the chosen saturation function outside of $\mathcal{X}$. However, due to the saturation, the broader space $\mathcal{O} + \delta_d$

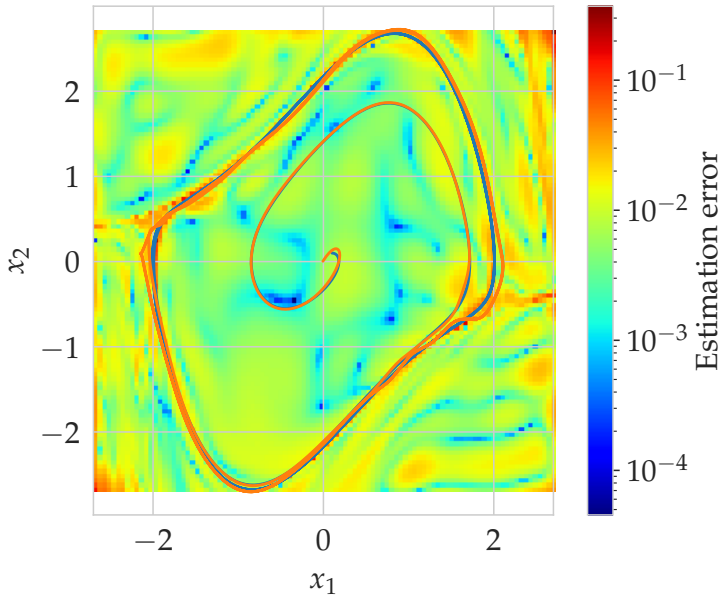


Figure 7.8: Heatmap of the state estimation error at $\omega_c = 0.2$ on a uniform grid over the training space $[-2.7, 2.7]^2$ for the saturated Van der Pol oscillator. The backward and forward sampling methodology enables a relatively homogenous approximation of $\mathcal{T}^*$ over the chosen state-space grid. A test trajectory for $x(0) = (0.1, 0.1)$ is also represented (true trajectory in blue, estimated in orange).

outside of which the dynamics are zero is backward invariant, such that there is a unique admissible transformation $\mathcal{T}$ that is valid on all of $\mathcal{O} + \delta_d$.

As in Sec. 7.4.1, we choose a set of hundred even-spaced values for ω_{c_i} in $[0.03, 1]$. Then, LHS is used to select $N = 5,000$ samples $x_i \in [-2.7, 2.7]^2$ for each value of ω_c, the corresponding z_i samples are computed with backward-forward sampling, D is set to (7.12), F to $(1, 1, 1)$, and t_c to $\frac{10}{\lambda_{min}(D)}$. The same NN model and loss function are used, with the same hyperparameters.

The empirical criterion is shown in Fig. 7.5. Again, we observe the trade-off between peaking represented in $|G_z|_{H^2}$ and smoothness represented in $|G_\epsilon|_\infty$ and in the gradient of $\mathcal{T}_\theta^*$. This is illustrated in Fig. 7.6 and 7.7. A heatmap of the estimation error on a uniform grid over $[-2.7, 2.7]^2$ is also represented in Fig. 7.8, along with a test trajectory, showing that the performance of the observer is relatively homogenous over the state-space thanks to backward-forward sampling.

7.4.3 Furuta pendulum

We then consider simulations of a rotational inverted pendulum or Furuta pendulum: the Qube Servo 2 by Quanser [204], previously studied in Ch. 6 and illustrated in Fig. 6.4. To generate the training data, we use the BlueRiverTech simulation model available online. In this chapter, we focus on learning KKL observers for autonomous systems. Hence, the scenarios of interest consist in lifting the pendulum to some initial position, then dropping it and recording the corresponding stabilizing trajectory. However, in this autonomous case, the provided simulation model reaches its limits. Indeed, many nonlinear terms such as damping and friction effects, which are not precisely known but negligible compared to the torque imposed by the motor when the pendulum is actuated and the oscillations are small, now become dominant. In particular, the value of θ_1 at the equilibrium is not determined by the

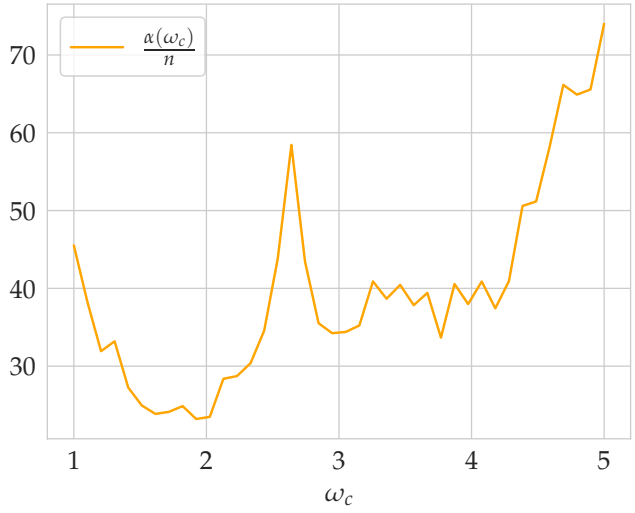


Figure 7.9: Proposed gain tuning criterion (7.10) for the pendulum, divided by $n = 50,000$. Choosing $\omega_c = 1.9$ appears to be optimal w.r.t. this metric.

equations of the dynamics. Therefore, the value at which it stabilizes is free according to the simulation model and only depends on unmodeled effects such as friction, electromagnetic damping, and the wire holding the rotary arm back. For example, for close enough initial conditions, the hardware system stabilizes at values of θ_1 close to zero, whereas these values are much further away in simulation. We conclude that the simulation model is highly sensitive to the values of the parameters and the initial condition, particularly for the value of θ_1 at the equilibrium, even more so in unactuated settings. To improve the accuracy of the simulations compared to the hardware data, we re-identify its parameters (motor resistance, back electromotive force constant, viscous damping coefficients in both joints) using a set of trajectories recorded on the hardware[5] and least-squares optimization on both outputs (θ_1, θ_2). The predictions of this updated model are closer to the experimental data than the original simulations. We use this somewhat improved model as our simulator for generating the training data.

We model this system with a state of dimension four, two angles (θ_1, θ_2) and two angular velocities $(\dot{\theta}_1, \dot{\theta}_2)$. We measure $y = \theta_1$; in this setting, the system is locally observable (the observation matrix of its linearized version has rank d_x) as discussed in Ch. 6. However, its trajectories diverge in finite backward time. Hence, as suggested in [47], [114], we consider the modified system

$$\begin{aligned} \dot{x} &= f(x)g(x), \\ g(x) &= \begin{cases} 1 & \text{if } |x| \leq r \\ 0 & \text{if } |x| \geq r + d \\ p(|x| - r) & \text{otherwise} \end{cases} \end{aligned} \tag{7.17}$$

where f is the dynamics model of the pendulum and g is a saturation function as in Sec. 7.4.2. We set $r = 50$ and $d = 100$. The function $p(\cdot)$ is a polynomial of order three chosen such that g be C^1.

Due to the curse of dimensionality, a large amount of data is necessary to learn

[5]Thanks to Sebastian Giedyk for his help collecting the experimental data.

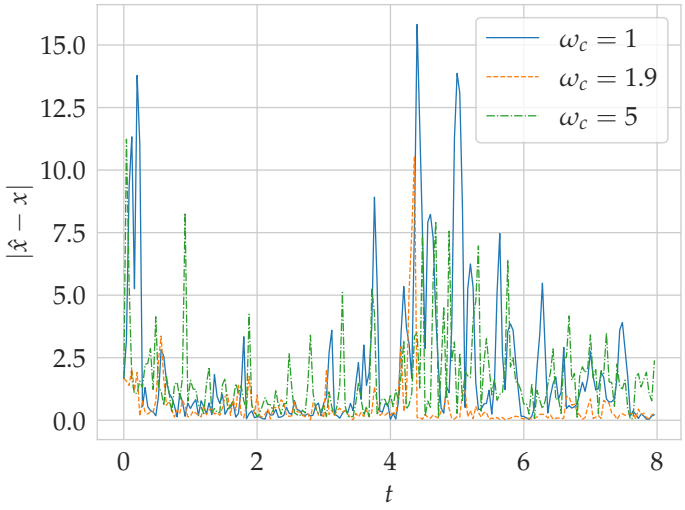
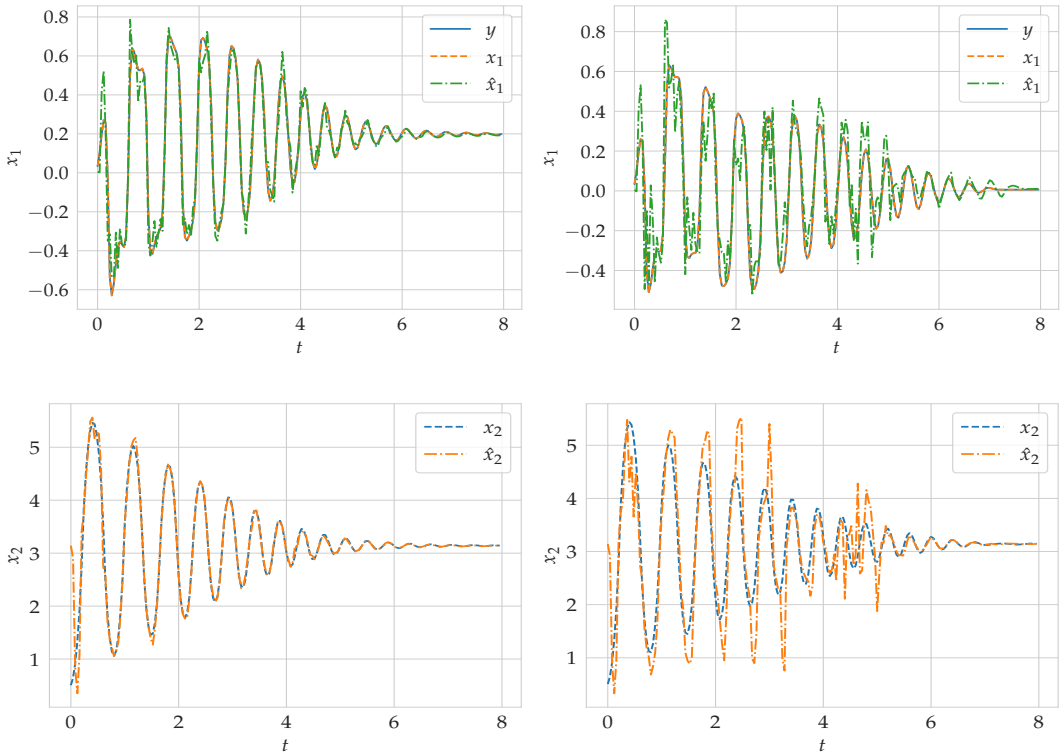


Figure 7.10: Estimation error for a simulated test trajectory of the pendulum starting at $x(0) = (0.1, 0.1, 0, 0)$, with Gaussian noise $\mathcal{N}(0, 0.025)$ on the measurement. We observe a high sensitivity to noise for high values of ω_c (RMSE $= 2.6$ for $\omega_c = 5$). In contrast, the sensitivity to noise is lower for low values of ω_c, but with long transients, which leads to RMSE $= 3.2$ for $\omega_c = 1$. The value $\omega_c = 1.9$ compromises between transient performance and sensitivity to noise (RMSE $= 0.9$).



(a) Simulation (b) Experiment

Figure 7.11: Estimated trajectories (θ_1 above, θ_2 below) of the Furuta pendulum with $\omega_c = 1.9$, without measurement noise. The trajectory obtained from simulation (left) leads to reasonable performance, while the one obtained from an experiment (right) is not smooth.

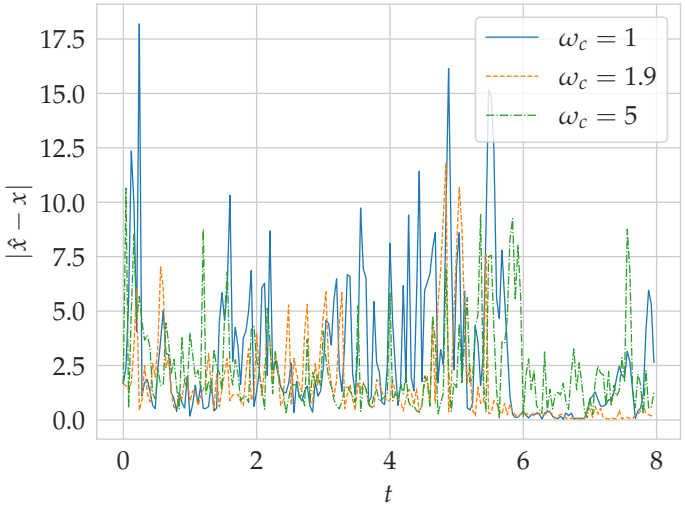


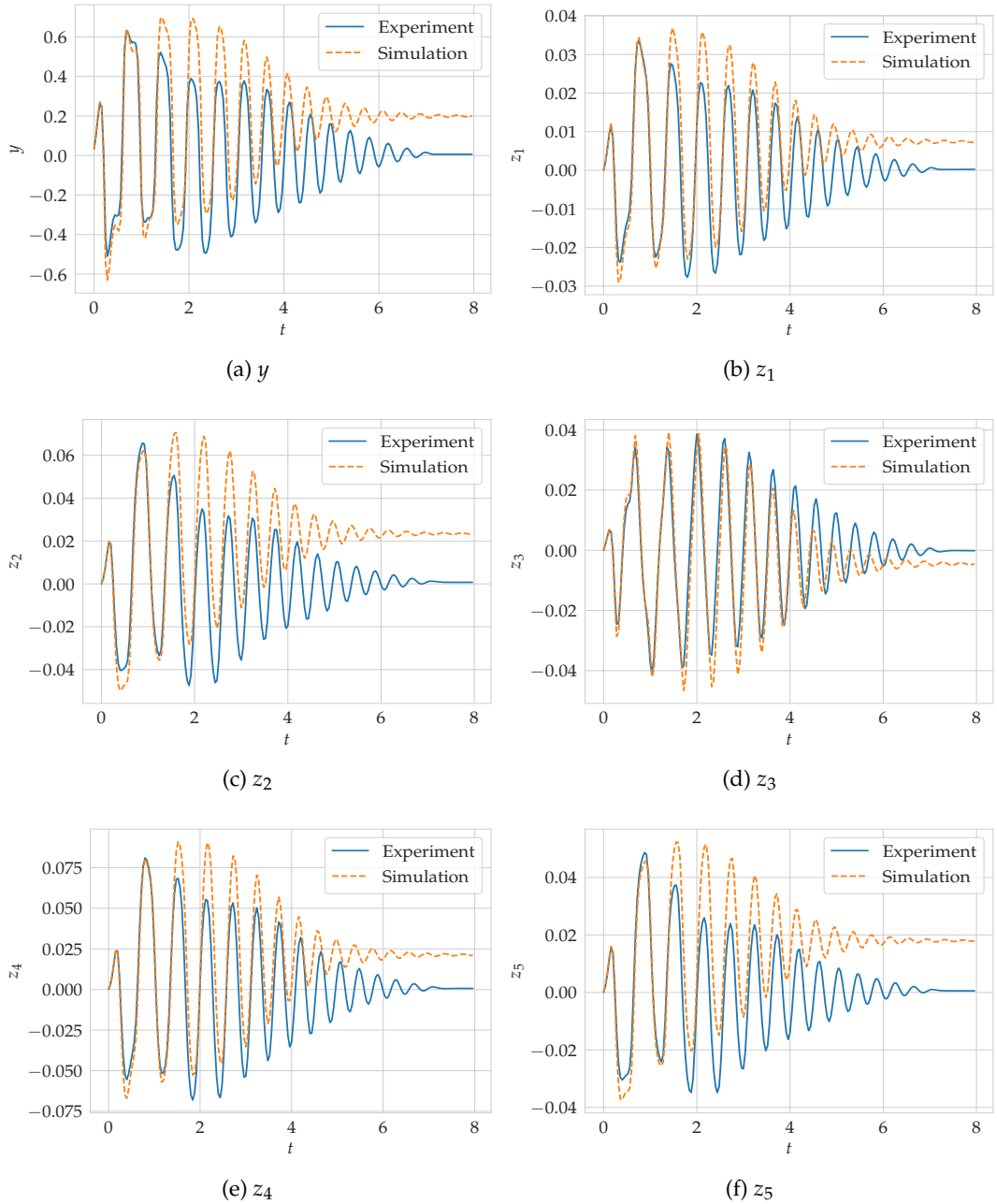
Figure 7.12: Estimation error $|\hat{x} - x|$ for an experimental test trajectory of the pendulum, recorded on the hardware, with Gaussian noise $\mathcal{N}(0, 0.025)$ on the measurement. The accuracy is lower due to model errors, however, choosing $\omega_c = 1.9$ again appears as a suitable compromise between the influence of measurement noise and the transient performance.

$\mathcal{T}^*$. Hence, we employ trajectory-based sampling as described in Sec. 7.3.2. We select 500 samples in the hypercube $[[-0.5, 0.5], [-0.5, 0.5], [-0.1, 0.1], [-0.1, 0.1]]$ around the upward equilibrium position, use backward-forward sampling to obtain the corresponding z values, then run a joint simulation of both the x and z trajectories for 8s, sampled with time steps of 0.04s. These trajectories then serve as training data, after remapping θ_1 to $[-\pi, \pi]$ and θ_2 to $[0, 2\pi]$ for continuous data. This leads to $N = 10^6$ points for each of 41 evenly spaced values of $\omega_c \in [1, 5]$.

Learning the transformation $\mathcal{T}^*$ as a function of ω_c as with the previous systems quickly becomes intractable for this higher-dimensional system. Hence, we learn an independent transformation for each considered value of ω_c, then compute the criterion (7.10) to choose a final model, as presented in option (ii) above.

We compute the empirical criterion (7.10) for each value of ω_c independently on a grid of $n = 50,000$ points and obtain Fig. 7.9. The criterion fluctuates locally due to the varying accuracy of the learned model for each ω_c. The minimum is reached at $\omega_c = 1.9$, which again seems to be a good compromise between long transients and sensitivity to measurement noise. This is illustrated in Fig. 7.10 on a simulated test trajectory: high values of ω_c lead to sensitivity to high-frequency measurement noise, low values to long transients whenever the estimate is off, and $\omega_c = 1.9$ to an acceptable trade-off.

We also test the learned observer on real test trajectories recorded on the hardware. The simulation model is inaccurate compared to the hardware data, as depicted in Fig. 7.13, where it is shown that the measurement and the observer state z differ for an experiment and the corresponding simulation. Therefore, the observer is bound to yield a higher estimation error on the experimental test trajectories. In particular, the estimated trajectory is not smooth, due to a combination of querying the observer outside of the training set, model errors in that training data, and overfitting. This is illustrated in Fig. 7.11. However, even if the accuracy of the estimated trajectory is rather low, its response to measurement noise is similar, and exhibits the same trade-offs as for simulation data. Hence, choosing $\omega_c = 1.9$ as per the gain tuning



(a) y (b) z_1

(c) z_2 (d) z_3

(e) z_4 (f) z_5

Figure 7.13: Measurement and state of the observer over time, without measurement noise, for an experimental trajectory collected on the hardware and the corresponding simulation.

criterion (7.10) appears to be the best possible compromise, as showcased in Fig. 7.12.

7.4.4 Conclusion on the supervised approach

These results constitute a first step towards gain tuning for nonlinear observers. They can be considered as a proof of concept, showing that it is possible to tune the gains of numerical KKL observers by parametrizing the gain matrix with a scalar ω_c then optimizing this scalar w.r.t. certain metrics. Note that many such metrics could be considered depending on the use case at hand. We propose the gain tuning criterion (7.10), which displays relevant aspects of the trade-off faced when choosing D as illustrated by our results, but other quantities could also be helpful.

7.5 Towards joint optimization

In the previous sections, we approximate the transformation $\mathcal{T}^*$ (and $\mathcal{T}$ if needed) using supervised learning. This approach does not permit optimizing D jointly with the transformations, as D needs to be fixed to build the dataset (x_i, z_i). To address this shortcoming, we propose another approach based on an unsupervised learning framework: autoencoders.

7.5.1 Autoencoders

Finding latent representations of the available data is an active research area in machine learning. In this regard, most state-of-the-art techniques are based on autoencoders (AE). These consist in two NNs that are trained jointly to learn the projection of an input into a latent space and back into the original space. Their aim is usually to find a meaningful, lower-dimensional representation of the data, in order to ease downstream tasks such as analysis or classification [220], to find a set of coordinates sufficient to predict the behavior of a system [29], [93], [104], or to generate synthetic samples [221].

Autoencoders can also embed an input into a space with some known structure. This structure can be probabilistic, as often for variational AEs [222], but can also derive from some known behavior of the latent space. For example, in [145], the authors propose an AE framework to learn a linear approximation of general nonlinear systems relying on the Koopman operator. The AE model approximates the projection from the original coordinates to the corresponding Koopman coordinates, in which the system behaves linearly, and back. The linear but infinite-dimensional Koopman operator that acts in these coordinates is approximated by a matrix jointly with the AE, by enforcing that the latent state follows linear dynamics driven by this operator.

Similarly to [145], we propose to learn $\mathcal{T}$ and $\mathcal{T}^*$ as a deterministic AE such that the latent variable z satisfies (7.2). As presented in Sec. 7.1.1 and contrarily to standard AE models, under the assumptions at hand, the existence of suitable $\mathcal{T}$ and $\mathcal{T}^*$ is guaranteed with a fixed dimension d_z and a fixed structure of the latent space. Note that this idea has recently been proposed in [117] for discrete-time KKL observers. We have investigated it in parallel, intending to compare supervised and unsupervised approaches and to optimize D jointly with the NNs.

The AE is trained on N samples x_i obtained from a chosen sampling method; there is no need to sample z, which eases the data generation process and significantly reduces the overall computational needs, as most of the computational power is spent on simulating the training data. It consists of two neural networks: an encoder and a

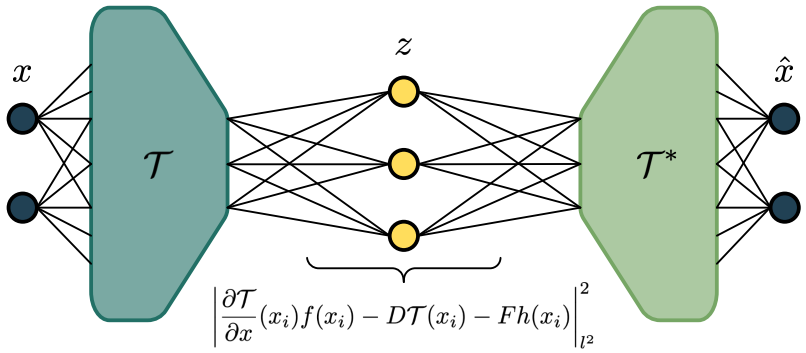


Figure 7.14: Structure of the autoencoder. The AE learns $\mathcal{T}, \mathcal{T}^*$ by minimizing the loss (7.18) made up of the PDE (7.4) and the reconstruction error between $\hat{x}$ and x.

decoder. The encoder $\mathcal{T}_\theta$ maps the input to the latent state $z = \mathcal{T}_\theta(x)$, and the decoder $\mathcal{T}_\theta^*$ maps it back to a reconstructed input $\hat{x} = \mathcal{T}_\theta^*(z)$. During training, the weights of both NNs concatenated into θ are updated to minimize the following cost function:

$$L_{AE}(\theta) = \frac{1}{2}\sum_{x_i} \lambda |x_i - \mathcal{T}_\theta^*(\mathcal{T}_\theta(x_i))|^2 + \left|\frac{\partial \mathcal{T}_\theta}{\partial x}(x_i)f(x_i) - D\mathcal{T}_\theta(x_i) - Fh(x_i)\right|^2. \quad (7.18)$$

The cost function is made up of two parts. First, the reconstruction loss, i.e., the mean squared error between x_i and $\mathcal{T}_\theta^*(\mathcal{T}_\theta(x_i))$, enforces $x \approx \mathcal{T}_\theta^*(\mathcal{T}_\theta(x))$. Second, the PDE (7.4) on $\mathcal{T}$ is enforced on the grid of x_i. Therefore, minimizing (7.18) boils down to approximating an invertible solution of (7.4). The loss terms are weighted by the scalar λ. The architecture of the AE model is illustrated in Fig. 7.14. To train this model, a grid of N data points x_i is generated with LHS or any other method, then the weights θ are optimized using gradient-based methods to minimize (7.18). Note that this type of loss function is closely related to physics-informed neural networks (Sec. 3.2.2); insights from this line of work could be used to improve the performance of this unsupervised method.

7.5.2 Jointly optimizing the gain matrix

One of the advantages of the AE model in Fig. 7.14 is that it enables optimizing D jointly with the network weights. This has already been implemented in the context of nonlinear system identification, e.g., in [159]. It can be done naturally by adding D to the model parameters when minimizing (7.18).

We train the AE in the same settings as the previous experiments on the reverse Duffing oscillator (7.13) with $\lambda = 0.1$ and $N = 70,000$. We initialize D as (7.12) with $\omega_c = 0.2$. After training, D has almost the same eigenvalues as at the initialization, and we observe similar performance to the previous supervised method, as illustrated in Fig. 7.15. Initializing with $\omega_c = 0.5$ slightly deteriorates the performance (RMSE = 0.075 on the test trajectory in Fig. 7.15) while the eigenvalues of D slightly decrease during training; they slightly increase when initializing with $\omega_c = 0.1$ (RMSE = 0.095). The results are depicted in Fig. 7.16. Looking at test trajectories with noise, similarly to Fig. 7.3, seems to indicate that $\omega_c \simeq 0.2$ is also optimal for the AE model in terms of robustness to measurement noise. Overall, the eigenvalues of D seem to move towards real values around $(-1, -2, -3)$, as seen in Fig. 7.16c, which has the best

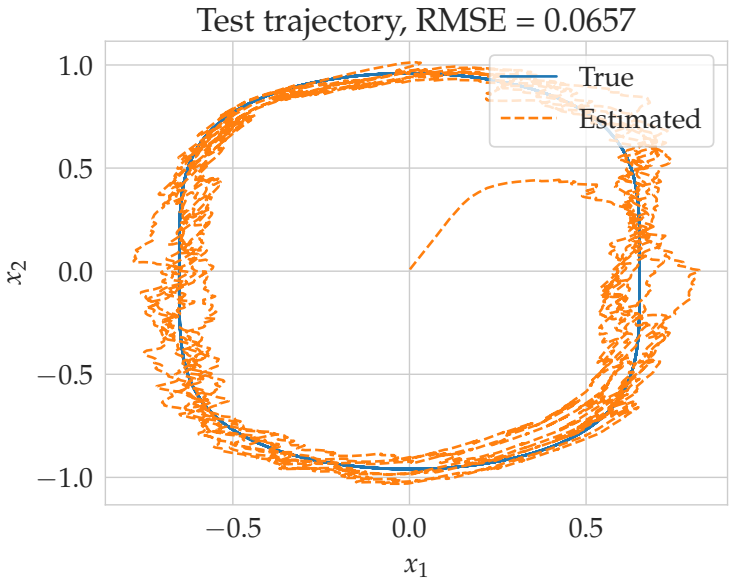


Figure 7.15: Estimated trajectory of the reverse Duffing oscillator starting at $x(0) = (0.6, 0.6)$ using an autoencoder and optimizing D jointly with the model weights. D is initialized as (7.12) with $\omega_c = 0.2$, and the measurement is corrupted by Gaussian noise $\mathcal{N}(0, 0.5)$ as in the bottom line of Fig. 7.3.

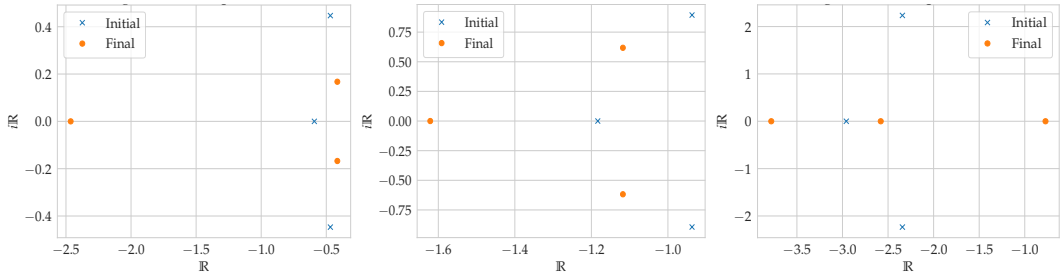
performance in terms of RMSE on test trajectories without noise. Setting D to the corresponding diagonal matrix indeed yields similar performance.

The value of D obtained by joint optimization in the unsupervised setting is different from a Bessel form, and could potentially lead to other parametrizations of D. We investigate this by taking the optimal value obtained for $\omega_c = 0.5$ (best performance without noise), denoted D_{opt} and shown in Fig. 7.16c, and run the supervised approach again with $D(k) = kD_{\text{opt}}$ where k is a free parameter. This new parametrization based on multiplying the gain matrix as optimized by the AE model by a factor leads to similar results as in Fig. 7.3. The corresponding gain criterion and a test trajectory corresponding to the bottom line of Fig. 7.3 are shown in Fig. 7.17. This again highlights that the parametrization of D is an important design choice, and that more experiments should be conducted to explore the different possibilities.

Training the same settings with the Van der Pol oscillator leads to similar performance as the supervised approach for high ω_c (around 1). However, for lower values, the transformation again seems more demanding to learn, and we do not manage to train it for $\omega_c < 0.5$. This can be attributed to the existence and injectivity of $\mathcal{T}$ being guaranteed only for high enough ω_c, such that the transformation may be less smooth for low ω_c and more difficult to approximate, even more so with unsupervised learning techniques which are widely recognized as more complicated to train.

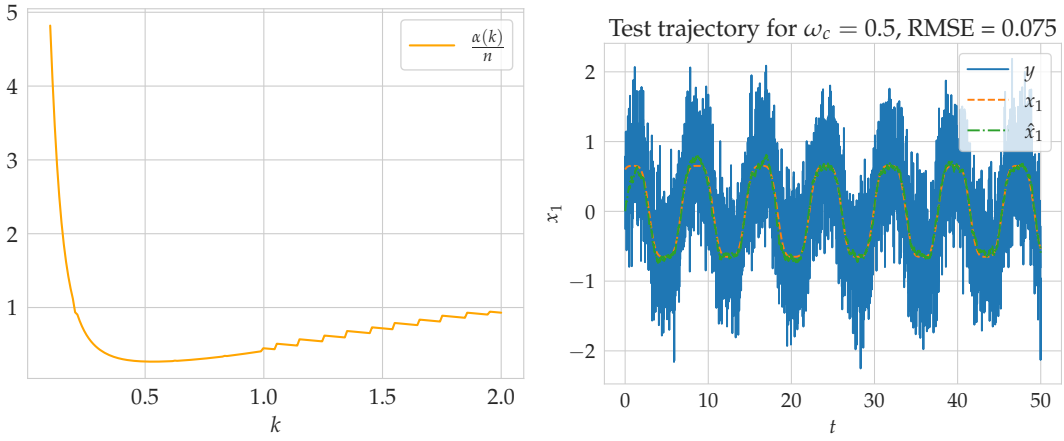
7.5.3 Conclusion on the unsupervised approach

These experiments demonstrate that it is possible to learn KKL observers by training an autoencoder from samples in $\mathcal{X}$ only, and to optimize the tuning parameters jointly with the model weights. With this unsupervised method also, initializing D as (7.12) with $\omega_c \approx 0.2$ yields good performance in the presence of noise.



(a) $\omega_c = 0.1$ (b) $\omega_c = 0.2$ (c) $\omega_c = 0.5$

Figure 7.16: Evolution of the eigenvalues of D for the reverse Duffing oscillator in the unsupervised setting. D is initialized as (7.12) for different values of ω_c, then optimized jointly with the AE.



(a) Gain criterion

(b) Test trajectory with $\omega_c = 0.5$

Figure 7.17: Supervised approach for the reverse Duffing oscillator, with D_{opt} obtained by joint optimization and multiplied by a factor k. The empirical gain criterion (left) is minimal for $k = 0.5$, which leads to a similar performance on a test trajectory (right) as $\omega_c = 0.15$ with the Bessel gain matrix in Fig. 7.3.

From then on, it is possible to add other terms to (7.18) to penalize other aspects, e.g., noise sensitivity, by adding our gain tuning criterion (7.10) which depends both on θ and on D to the loss. This could be a good direction for future research, but requires significant implementation efforts to implement cost functions resembling (7.10) in automatic differentiation software such as `PyTorch`. Therefore, we leave this open for future work, once more control-theoretic or signal analysis functions are added to such software.

7.6 Conclusion and outlook

In this chapter, we tackle the problem of gain tuning for KKL observers of autonomous nonlinear systems. We propose to numerically approximate the observer from simulation data, as introduced in [116], with an improved backward-forward sampling scheme. We parametrize the observer dynamics matrix D with a scalar ω_c, derive an empirical criterion for tuning it, and demonstrate on numerical examples that it

encompasses some relevant aspects of its influence on the performance. We propose to either learn an observer for each value of ω_c of interest, or to directly learn a family of models that also takes this parameter as an input.

Similarly to [117], [145], it is also possible to learn a model of $\mathcal{T}$ and $\mathcal{T}^*$ jointly using an autoencoder structure, such that the latent variable z satisfies (7.2). The cost function is then made up of a reconstruction loss and a loss on the PDE (7.4) satisfied by $\mathcal{T}$, such that an invertible solution to (7.4) is approximated on a grid of samples of x. This approach enables the user to optimize D jointly with the models $\mathcal{T}_\theta$, $\mathcal{T}_\theta^*$ and to add terms to the cost functions to penalize other aspects, such as the criterion (7.10). However, it is also more difficult to train than the supervised approach.

Further recent research aims at improving the accuracy of learning-based KKL observers, which could be combined with the insights provided in this chapter for future use. For example, in [118], the supervised and unsupervised settings proposed above are mixed to learn a PINN: both $\mathcal{T}_\theta$ and $\mathcal{T}_\theta^*$ are learned jointly by minimizing the AE loss (7.18), but supervised training points (x_i, z_i) are generated and a term in $|z_i - \mathcal{T}_\theta(x_i)|$ is added to the AE loss. This problem formulation, along with other practical considerations, lead to more accurate results than the supervised or unsupervised methods taken separately. In [223], the learning problem is formulated with NODEs, so that the parameters θ are updated to fit the trajectory of $\hat{x} = \mathcal{T}_\theta^*(z)$, with $\dot{z} = D_\theta z + Fy$, to a true trajectory of the system. D_θ is optimized jointly and noise is added to the training data for more robustness. This also leads to satisfactory results for the resulting numerical KKL.

Many questions remain open. As often in machine learning, it is unclear how to sample the state-space to optimally generate the training set. Iterative active learning procedures can be envisioned, for example by learning the observer, then resampling in the parts of the state-space with the highest error, and learning again until the desired accuracy is achieved everywhere. Such mesh refinement techniques [224] based on a performance criterion, e.g., the gradient of $\mathcal{T}^*$ or the estimation error, could help improve the performance of the obtained observer. However, they require computing $\mathcal{T}^*$ at many points, which can be sped up e.g., with surrogate models [225], over a grid of z_i points which is necessarily irregular since they correspond to the originally chosen x_i points. Each new grid and corresponding model are also expensive to compute. Hence, these considerations seem to induce extensive computations for limited gain. Selecting the state-space grid a priori to achieve a given accuracy on the transformations could also be considered, as investigated in [218], but assumptions on the smoothness of $\mathcal{T}^*$ are needed.

Extending KKL observers to nonautonomous systems is investigated in [114]; adapting the learning-based methodology to such systems is also a topic for future research. First attempts are proposed in [116], [223] for control-affine systems, for which the transformation along an unseen control input can be derived from the transformation learned with a given input trajectory. Another promising path could be functional observers [115], where an observable function of the state is estimated directly. This can be useful for output feedback, but also for extending the method to nonlinear systems: if the input can be modeled by an auxiliary dynamical system, then a functional observer can be built for the original state, i.e., a part of this extended system, as a numerical KKL. However, this requires also sampling the auxiliary state that generates the output; the curse of dimensionality and the resulting amount of necessary data could hinder such approaches.

Analyzing the performance of the learned observer is also nontrivial. When computing the estimation error over the state-space, a given error level in a given region can either be caused by approximation errors (a purely numerical issue) or

by reduced observability in that region. It would be useful to understand how to disentangle these two aspects, such that the performance of the approximate observer can be characterized a priori. The methods investigated in Chapter 6 could be a first step, as they enable analyzing distinguishability quantitatively in the state-space using output data. Comparing the heatmap of the estimation error with a given observer and the distinguishability heatmap should reveal in which regions the observer could be tuned to reach high estimation accuracy, and in which regions this is unlikely due to low distinguishability.

Chapter 8
Conclusion - version française

Les applications de l'apprentissage automatique aux secteurs industriels sont porteuses de nombreuses promesses, allant d'éviter les temps de panne grâce à la maintenance prédictive à l'automatisation de tâches complexes à l'aide de capteurs bon marché et robustes. La combinaison de ces techniques avec les outils classiques de la théorie des systèmes et de la simulation numérique ouvre de nouvelles portes, comme la possibilité de créer des jumeaux digitaux, c'est-à-dire des répliques numériques de systèmes physiques. Pour ce faire, la première étape nécessaire est de construire des modèles de simulation fonctionnant presque en temps réel, par exemple en apprenant des modèles d'ordre réduit avec Ansys DynaROM. Ensuite, le modèle numérique peut être affiné à l'aide de données expérimentales provenant du système physique. Plus généralement, il y a un intérêt croissant pour l'exploitation des données générées par les plateformes physiques, tout en utilisant la compréhension de la théorie des systèmes. Dans cette thèse, nous développons des méthodes génériques pour extraire des informations des données expérimentales générées par un système dynamique.

Résumé

Dans la première partie, nous nous concentrons sur l'amélioration des modèles sous forme d'état (SSM) avec des observations partielles. Dans le Ch. 4, nous supposons une forme canonique observable spécifique et nous interconnectons un observateur grand gain (HGO) avec un modèle de processus gaussien discret, qui est régulièrement mis à jour avec les trajectoires d'état complètes estimées par l'observateur. En tirant parti de la robustesse des observateurs grand gain aux erreurs de modèle, nous prouvons la convergence conjointe de l'état estimé et du modèle dynamique. Cette nouvelle approche permet d'associer des garanties théoriques et des capacités de modélisation universelles. Remplacer le HGO par un filtre de Kalman étendu (EKF) permet d'étendre la méthodologie à des représentations générales de l'espace d'état, à condition que le modèle a priori soit raisonnablement précis. Cependant, les HGO et les EKF présentent tous deux des limites pratiques, telles que la robustesse au bruit de mesure et à l'erreur de modèle. C'est pourquoi nous proposons une formulation plus générale dans le Ch. 5, basée sur des équations différentielles ordinaires neuronales (NODEs). Ce cadre flexible permet d'appliquer un large éventail de connaissances préalables sur le SSM sous-jacent. En le combinant avec un modèle de reconnaissance basé sur les observateurs de Kazantis-Kravaris / Luenberger (KKL) pour faire correspondre les mesures à l'état latent initial de la NODE, on obtient une méthodologie "end-to-end" pour l'apprentissage de la dynamique à partir d'observations partielles avec différents degrés de connaissances préalables. Les méthodes proposées sont générales et conduisent à des performances acceptables compte tenu des paramètres donnés

Dans la deuxième partie, nous passons de l'identification du système à l'estimation

de l'état. Dans le Ch. 6, nous examinons comment mesurer l'observabilité des systèmes stochastiques non linéaires. Nous définissons la distinguabilité distributionnelle pour de tels systèmes comme le fait d'avoir des distributions de sortie différentes, et nous montrons qu'elle étend la notion déterministe. Nous proposons ensuite une nouvelle quantification de la distinguabilité distributionnelle à partir des données de sortie, en utilisant des méthodes à noyaux pour comparer les distributions à partir d'échantillons. Nous appliquons également un test statistique pour déterminer à partir de quel seuil deux distributions initiales peuvent être considérées comme distinctes avec un niveau de confiance élevé. Dans le chapitre Ch. 7, nous étudions le design d'observateurs KKL pour les systèmes non linéaires généraux. Nous nous appuyons sur les observateurs KKL dits numériques, où la transformation de l'observateur aux coordonnées physiques est apprise à partir de données de simulation, et nous proposons un premier critère empirique pour calibrer la matrice de gain libre. Les deux méthodes réduisent un problème de dimension infinie, c'est-à-dire l'évaluation de l'observabilité ou le calcul d'une transformation entre des représentations appropriées de l'espace d'état, à de nombreuses simulations numériques et des approches "brute force", c'est-à-dire la comparaison de distributions ou l'apprentissage d'une approximation basée sur des échantillons. Cela permet de résoudre les problèmes considérés en faible dimension.

Discussion

Les approches résumées ci-dessus ont été obtenues en combinant des résultats théoriques de théorie du contrôle avec des concepts et des outils d'apprentissage automatique, afin d'extraire des informations des données expérimentales sur le SSM sous-jacent. L'association de ces deux points de vue permet d'aborder de nouveaux problèmes et conduit à de nouveaux résultats. Les preuves de concept fournies dans cette thèse sont suffisamment génériques pour être adaptées à de nombreux cas d'usage, cependant, davantage de travail théorique et de connaissances pratiques sont nécessaires pour décider comment les adapter et maximiser les performances.

En particulier, la plupart des techniques d'apprentissage automatique et de théorie du contrôle nécessitent, en pratique, une quantité importante de calibration et de réglages. Les outils étudiés dans cette thèse automatisent certains aspects en favorisant des méthodes génériques, en s'appuyant sur des modèles universels tels que les GP avec des noyaux exponentiels ou en ayant une dimension fixe pour l'état latent dans les modèles de reconnaissance KKL. Cependant, elles héritent toujours du besoin de calibration : l'architecture du réseau neuronal, les hyperparamètres du noyau, le taux d'apprentissage, les gains de l'observateur, etc, jouent tous un rôle important, mais il n'est pas simple de les définir. Nous proposons une telle heuristique pour le réglage des observateurs KKL numériques dans le Ch. 7, mais d'autres choix tels que la forme de la matrice de gain restent non spécifiés. L'exécution des méthodes proposées dans la pratique nécessite souvent une compréhension et des astuces spécifiques à chaque cas.

Des travaux théoriques sont donc encore nécessaires pour comprendre et prédire quand ces méthodes échoueront. C'est particulièrement vrai lorsqu'on utilise des modèles d'apprentissage profond, dont les performances sont encore difficiles à prévoir. D'autres domaines de l'apprentissage automatique, tels que les méthodes à noyau, ont récemment montré leur applicabilité pratique tout en étant théoriquement bien fondées, et en permettant plus facilement une analyse et des garanties analytiques. L'extension de ces méthodes aux problèmes étudiés dans cette thèse,

tels que l'apprentissage des résidus d'un modèle préalable à partir d'observations partielles comme dans le Ch. 5 ou la distinction entre des distributions à partir de données comme dans le Ch. 6, pourrait être une perspective intéressante. Par exemple, l'élaboration d'un théorème du représentant pour apprendre les SSM à partir d'observations, comme indiqué dans la Sec. 5.7, pourrait conduire à des modèles applicables en pratique et assortis de garanties.

Au cours de cette thèse, nous avons pu constater que la collecte de données physiques est un processus coûteux. C'est pourquoi la construction de simulateurs haute fidélité pour générer des données virtuelles ou pour les utiliser en tant que modèles a priori est d'un grand intérêt. Néanmoins, les besoins informatiques restent des problèmes majeurs pour la plupart des méthodes basées sur l'apprentissage. Bien que de nombreux outils modernes soient efficaces d'un point de vue informatique (par exemple, l'entraînement rapide de réseaux de neurones par des méthodes de descente de gradient stochastique), la génération, la collecte et la manipulation de données passent toujours très mal à l'échelle. Par exemple, l'apprentissage supervisé d'un observateur numérique KKL devient impossible pour $d_x > 4$ environ, en raison de la nécessité d'échantillonner $\mathbb{R}^{d_x}$ et $\mathbb{R}^{d_z}$. Il est également important de se demander si la quantité de calcul et l'utilisation associée des ressources nécessaires à l'exécution de ces méthodes valent le gain de performance obtenu. Par exemple, l'entraînement d'une NODE avec un modèle de reconnaissance est un processus intensif en termes de ressources informatiques et humaines ; il ne devrait être entrepris que si des modèles plus simples ne peuvent être utilisés et si le gain attendu d'un modèle dynamique plus précis le justifie.

Nous avons également eu l'occasion d'appliquer ces idées à des cas d'usage plus industriels, tels que les données Fluent dans le Ch. 4 ou les données Wandercraft dans le Ch. 5. Ces cas plus complexes ont montré qu'il est nécessaire d'adapter nos approches générales non seulement au système en question, mais aussi aux objectifs du modèle. Par exemple, la calibration des pôles d'un observateur numérique KKL, comme dans le Ch. 7, implique de connaître les niveaux de bruit susceptibles d'être rencontrés et les priorités fixées. Dans le Ch. 7, nous proposons un critère empirique qui établit un compromis entre la sensibilité au bruit de mesure et les performances transitoires, mais d'autres cas d'usage pourraient conduire à d'autres préférences et donc à d'autres critères. Les futurs utilisateurs devront donc réfléchir au type d'informations qu'ils cherchent à extraire de leurs données expérimentales et à la manière dont le modèle résultant sera utilisé.

Perspectives

Les travaux menés dans le cadre de cette thèse peuvent être étendus selon deux voies principales. D'une part, on pourrait exploiter des résultats théoriques généraux pour améliorer la performance des méthodes combinant théorie du contrôle et apprentissage automatique. Par exemple, des résultats d'optimisation peuvent être mis à profit pour améliorer l'identification des systèmes formulée dans la première partie de la thèse. Par exemple, il est démontré dans [151] que les longues trajectoires peuvent faire diverger la fonction de coût dans de tels problèmes ; par conséquent, des trajectoires plus courtes conduisent à des conditions plus réalisables pour l'identification du système. Ces connaissances sont utilisées dans le Ch. 5. Les résultats théoriques sur les observateurs KKL justifient également les modèles de reconnaissance proposés dans le Ch. 5 et les observateurs numériques du Ch. 7. D'autres résultats pourraient être utilisés pour concevoir des stratégies d'échantillonnage plus informatives,

e.g., des garanties d'approximation sur la finesse de la grille nécessaire pour approximer une transformation selon son degré de continuité, comme dans [218]. Il existe peu de résultats théoriques décrivant les performances des méthodes d'apprentissage profond pour des applications en théorie du contrôle [38], [39] ; ils pourraient être étendus, ou d'autres approches facilitant l'analyse théorique, telles que les méthodes à noyaux, pourraient être envisagées.

D'autre part, il est possible de se concentrer sur des types particuliers de systèmes et de tirer parti de leurs caractéristiques spécifiques. Les systèmes présentant des invariances, par exemple conservant une fonction hamiltonienne le long des trajectoires, ont fait l'objet d'une grande attention, comme nous l'avons vu au Ch. 5. D'autres invariances, telles que les symétries, pourraient être exploitées, par exemple en construisant des observateurs KKL invariants. Ceux-ci pourraient être formés dans une partie de l'espace d'état comme dans Ch. 7, puis étendus à d'autres parties grâce à leur invariance. Connaître les propriétés de stabilité d'un système peut également faciliter son identification, comme cela a été examiné dans [78], [164], et pourrait être combiné avec les considérations du Ch. 5.

Tout au long de cette thèse, nous nous efforçons de développer des méthodes génériques qui peuvent être adaptées à de nombreux cas d'usage, en laissant de la place à la flexibilité. Par exemple, la combinaison d'un HGO et d'un GP proposée dans le Ch. 4 peut être utilisée pour apprendre les résidus d'un modèle a priori, ou les modèles NODE proposés dans le Ch. 5 peuvent être adaptés à un grand spectre de connaissances physiques. Tout le code nécessaire pour reproduire les résultats est open source et disponible en ligne ; voir la Sec. 2.2 pour une description détaillée. Nous nous sommes efforcés de rendre ce code modulaire et facile à manipuler pour de futurs utilisateurs. Nous encourageons donc tout lecteur à télécharger le code et à utiliser les leviers fournis pour l'adapter à son propre cas d'usage.

Chapter 9
Conclusion

Applications of machine learning to industrial sectors carry many promises, from avoiding breakdown time through predictive maintenance to automatizing complex tasks based on cheap and robust sensors. Combining these techniques with classical tools from system theory and numerical simulation opens new doors, such as the possibility to create digital twins, i.e., numerical replicas of physical systems. For this, the first necessary step is to build simulation models running nearly in real-time, e.g., by learning reduced order models with Ansys DynaROM. Then, the numerical model can be refined given experimental data from the physical system. More generally, there is a rising interest in leveraging the data generated by physical platforms, while making use of system theoretic understanding. In this thesis, we *develop generic methods for extracting information from experimental data generated by a dynamical system*.

Summary

In the first part, we focus on improving state-space models (SSMs) with partial observations. In Ch. 4, we assume a specific observable canonical form, and interconnect a high-gain observer (HGO) with a discrete Gaussian process (GP) model, which is regularly updated with the full state trajectories estimated by the observer. Leveraging the robustness to model errors of HGOs, we prove joint convergence of both the estimated state and dynamics model. This novel approach enables associating theoretical guarantees and universal modeling capabilities. Replacing the HGO with an extended Kalman filter (EKF) extends the methodology to general state-space representations given a reasonably accurate prior model. However, both HGOs and EKFs display practical limitations, such as robustness to measurement noise resp. model error. Therefore, we propose a more general formulation in Ch. 5 based on neural ordinary differential equations (NODEs). This flexible framework can enforce a wide spectrum of prior knowledge on the underlying SSM. Combining it with a recognition model based on Kazantis-Kravaris / Luenberger (KKL) observers to map the measurements to the initial latent state of the NODE yields an end-to-end framework for learning dynamics from partial observations with varying degrees of prior knowledge. The proposed methods are general and lead to acceptable performance given the wide settings.

In the second part, we shift the focus from system identification to state estimation. In Ch. 6, we examine how to measure the observability of nonlinear stochastic systems. We define distributional distinguishability for such systems as having different output distributions, and show it extends the deterministic notion. We then propose a novel quantification of distributional distinguishability from output data, making use of kernel methods to compare distributions from samples. We also apply a statistical test to determine above which threshold two initial distributions can be considered distinguishable with high confidence. In Ch. 7, we investigate designing KKL observers for general nonlinear systems. We build on so-called numerical KKL observers, where

the transformation from the observer to the physical coordinates is learned from simulation data, and propose a first empirical criterion to tune the free gain matrix. Both methods reduce an infinite dimensional problem, i.e., assessing observability or computing a transformation between suitable state-space representations, to many numerical simulations and brute force approaches, i.e., comparing distributions or learning an approximation based on samples. This renders the problems feasible in low dimensions.

Discussion

The approaches summarized above have been obtained by combining theoretical results from control theory with concepts and tools from machine learning, in order to extract information from experimental data about the underlying SSM. Joining these two views enables tackling new problems and leads to new results. The proofs of concept provided in this thesis are generic enough to be adapted to many use cases, however, more theoretical work and practical insight are necessary to decide how to adapt them and maximize performance.

In particular, most techniques in both machine learning and control theory require a significant amount of tuning in practice. The tools investigated in this thesis automate some aspects by favoring generic methods, e.g., relying on universal models such as GPs with squared exponential kernels or having a fixed dimension for the latent state in KKL recognition models. However, they still inherit the need for tuning: the architecture of the neural network, hyperparameters of the kernel, learning rate, gains of the observer, etc, all play an important role, yet setting them is not straightforward. Hence, heuristics that guide the choice of such parameters are still needed; we propose such a heuristic for tuning numerical KKL observers in Ch. 7, but other design choices such as the form of the KKL gain matrix remain unspecified. Executing the proposed methods in practice often requires case-specific understanding and "tricks".

Thus, theoretical work is still needed to understand and predict when such methods will fail. This is especially true when utilizing deep learning models, whose performance is still difficult to foretell. Other fields of machine learning such as kernel methods have recently shown their practical applicability while being theoretically well-founded, and more easily allowing for analytical analysis and guarantees. Extending these methods to the problems investigated in this thesis, such as learning the residuals of a prior model from partial observations as in Ch. 5 or telling distributions apart from data as in Ch. 6, could be an interesting perspective. For example, deriving a representer theorem for utilizing kernel Ridge regression to learn SSMs from observations as discussed in Sec. 5.7 could lead to practically applicable models that bear guarantees.

Over the course of this thesis, we have experienced firsthand that collecting physical data is an expensive process. Thus, building high-fidelity simulators to generate virtual data or to be used as priors is of major interest. Nonetheless, scalability and computational needs remain major issues for most learning-based methods. Though many modern tools are computationally efficient (e.g., the fast training of neural networks through stochastic gradient descent methods), data generation, collection, and manipulation are still strongly subject to the curse of dimensionality. For example, training a numerical KKL observer in a supervised fashion becomes intractable for $d_x > 4$ approximately, due to the need to sample $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_z}$. Whether the amount of computation and associated usage of resources necessary to run these methods are

worth the obtained performance gain is also an open question. For instance, training an NODE with a recognition model is an intensive process in both computational and human resources; it should only be attempted if simpler models cannot be used, and if the gain expected from a more accurate dynamics model justifies it.

We have also had the opportunity to apply our ideas to more industrial use cases, such as the Fluent dataset in Ch. 4 or the Wandercraft dataset in Ch. 5. These have shown that larger systems require tailoring our broad approaches not only to the system at hand, but also to the objectives of the model. For example, tuning the poles of a numerical KKL observer as in Ch. 7 involves knowing the noise levels that may be encountered and the priorities that may be set. In Ch. 7, we propose an empirical criterion that trades off sensitivity to measurement noise and transient performance, but other use cases could lead to other preferences and therefore other criteria. Hence, future users should reflect on the type of information they may seek to extract from their experimental data, and on how the resulting model may be used.

Perspectives

The work conducted in this thesis can be further extended by pursuing two main paths. On the one hand, one could exploit general theoretical results to enhance the performance of methods combining control theory and machine learning. For example, findings in optimization can be leveraged to improve system identification as formulated in the first part of the thesis. It is shown in [151] that long trajectories can lead the objective function in such problems to diverge; hence, shorter trajectories lead to more feasible conditions for optimization-based system identification. This knowledge is utilized in Ch. 5. Theoretical results on the KKL observer also enable the recognition models proposed in Ch. 5 and the numerical observers from Ch. 7, e.g., having an upper bound for the dimension d_z of the observer state and knowing the transformation $\mathcal{T}$ is injective. Other findings could be drawn upon to design more informative sampling strategies, e.g., approximation guarantees on the fineness of the grid necessary to approximate a transformation based on its smoothness, as in [218]. There are few theoretical results to describe the performance of deep learning methods for applications in control theory [38], [39]; they could be further extended, or other approaches facilitating the theoretical analysis, such as kernel methods, could be envisioned.

On the other hand, one could focus on particular types of systems and leverage their specific characteristics. Systems with invariances, e.g., that conserve a Hamiltonian function along trajectories, have drawn great attention, as discussed in Ch. 5. Other invariances such as symmetries could be exploited, for example by building invariant KKL observers. These could be trained in a part of the state-space as in Ch. 7, then extended to other parts thanks to their invariance. Knowing the stability properties of a system can also ease its identification, as examined in [78], [164], and could be combined with the considerations in Ch. 5.

Throughout this thesis, we attempt to develop generic methods that can be adapted to many use cases, by leaving room for flexibility. For example, the combination of HGO and GP model proposed in Ch. 4 can be used to learn the residuals of a prior model, or the NODE models proposed in Ch. 5 can enforce a wide spectrum of prior knowledge. All code needed to reproduce the results is open source and available online; see Sec. 2.2 for a detailed description. We strive to make this code modular and easy to manipulate for future users. Therefore, we encourage any reader

to download the code and make use of the provided flexibilities to adapt it to their own use case.

Bibliography

[1] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[2] H. van der Valk, H. Haße, F. Möller, M. Arbter, J. L. Henning, and B. Otto, "A taxonomy of digital twins," in *Proceedings of the 26th Americas Conference on Information Systems*, 2020.

[3] F. Tao and Q. Qi, "Make More Digital Twins," *Nature*, vol. 573, no. 7775, pp. 490–491, 2019.

[4] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[5] A. Aublet, F. N'Guyen, H. Proudhon, and D. Ryckelynck, "Multimodal data augmentation for digital twining assisted by artificial intelligence in mechanics of materials," *Frontiers in Materials*, vol. 9, 2022.

[6] A. Dubey, R. Relan, U. Lohse, and J. Szwedowicz, "A Nonlinear Dynamic Reduced-Order Model for a Large Gas Turbine Outer Casing Low Cycle Fatigue Prediction," in *Proceedings of the ASME Gas Turbine India Conference*, 2021.

[7] J. Tomasi, F. Le Bars, C. Shao, *et al.*, "Patient-specific and real-time model of numerical simulation of the hemodynamics of type B aortic dissections," *Medical Hypotheses*, vol. 135, p. 109 477, 2020.

[8] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck, "Data Augmentation and Feature Selection for Automatic Model Recommendation in Computational Physics," *Mathematical and Computational Applications*, vol. 26, no. 1, p. 17, 2021.

[9] T. Daniel, F. Casenave, N. Akkari, A. Ketata, and D. Ryckelynck, "Physics-informed cluster analysis and a priori efficiency criterion for the construction of local reduced-order bases," *Journal of Computational Physics*, vol. 458, 2022.

[10] G. El Haber, J. Viquerat, A. Larcher, *et al.*, "Deep learning model to assist multiphysics conjugate problems," *Physics of Fluids*, vol. 34, no. 1, 2022.

[11] T. Wang, X. Bao, I. Clavera, *et al.*, "Benchmarking model-based reinforcement learning," *Preprint arXiv:1907.02057*, pp. 1–25, 2019.

[12] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *International Journal of Robotics Research*, pp. 1–24, 2021.

[13] A. Fabisch, C. Petzoldt, M. Otto, and F. Kirchner, "A Survey of Behavior Learning Applications in Robotics - State of the Art and Perspectives," *Preprint arXiv:1906.01868*, pp. 1–38, 2018.

[14] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: A survey," *Cognitive Processing*, vol. 12, pp. 319–340, 2011.

[15] M. Viberg, "Subspace-based methods for the identification of linear time-invariant systems," *Automatica*, vol. 31, no. 12, pp. 1835–1851, 1995.

[16] J. Schoukens and L. Ljung, "Nonlinear System Identification: A User-Oriented Roadmap," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 28–99, 2019.

[17] L. Ljung, *System Identification: Theory for the User*. Englewood Cliffs, New Jersey: Prentice Hall PTR, 1987.

[18] K. Schittkowski, *Numerical Data Fitting in Dynamical Systems*. Springer, Boston, MA, 2002.

[19] A. F. Villaverde, D. Pathirana, F. Fröhlich, J. Hasenauer, and J. R. Banga, "A protocol for dynamic model calibration," *Preprint arXiv:1902.11136*, 2021.

[20] A. Raue, M. Schilling, J. Bachmann, *et al.*, "Lessons Learned from Quantitative Dynamical Modeling in Systems Biology," *PLoS ONE*, vol. 8, no. 9, 2013.

[21] N. Galioto and A. A. Gorodetsky, "Bayesian system ID: optimal management of parameter, model, and measurement uncertainty," *Nonlinear Dynamics*, vol. 102, pp. 241–267, 2020.

[22] A. Wigren, J. Wågberg, F. Lindsten, A. G. Wills, and T. B. Schön, "Nonlinear System Identification: Learning While Respecting Physical Models Using a Sequential Monte Carlo Method," *IEEE Control Systems Magazine*, vol. 42, no. 1, pp. 75–102, 2022.

[23] J. Sjöberg, Q. Zhang, L. Ljung, *et al.*, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[24] J. Kocijan, *Modelling and Control of Dynamic Systems Using Gaussian Process Models* (Advances in Industrial Control). Springer International Publishing, 2016.

[25] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.

[26] J. Umlauft, "Safe Learning Control for Gaussian Process Models," Ph.D. dissertation, Technische Universität München, 2020.

[27] A. A. Ahmadi, A. Chaudhry, V. Sindhwani, and S. Tu, "Safely learning dynamical systems from short trajectories," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, vol. 144, 2021, pp. 498–509.

[28] K. Champion, B. Lusch, J. Nathan Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 45, pp. 22 445–22 451, 2019.

[29] J. Bakarji, K. Champion, J. N. Kutz, and S. L. Brunton, "Discovering Governing Equations from Partial Measurements with Deep Delay Autoencoders," *Preprint arXiv:2201.05136*, 2022.

[30] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "HiPPO: Recurrent memory with optimal polynomial projections," in *Advances in Neural Information Processing Systems*, 2020.

[31] A. P. Trischler and G. M. D'Eleuterio, "Synthesis of recurrent neural networks for dynamical system simulation," *Neural Networks*, vol. 80, pp. 67–78, 2016.

[32] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," in *Advances in Neural Information Processing Systems*, 2018, 6572–6583.

[33] S. Massaroli, M. Poli, J. Park, A. Yamashita, and H. Asama, "Dissecting neural ODEs," in *Advances in Neural Information Processing Systems*, 2020, pp. 3952–3963.

[34] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis, "On learning Hamiltonian systems from data," *Chaos*, vol. 29, no. 12, 2019.

[35] C. M. Legaard, T. Schranz, G. Schweiger, *et al.*, "Constructing Neural Network-Based Models for Simulating Dynamical Systems," *ACM Computing Surveys*, vol. 1, no. 1, 2021.

[36] R. Wang and R. Yu, "Physics-Guided Deep Learning for Dynamical Systems: A survey," *Preprint arXiv:2107.01272*, 2021.

[37] S. A. Faroughi, N. M. Pawar, C. Fernandes, S. Das, N. K. Kalantari, and S. Kourosh Mahjour, "Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks in Scientific Computing," *Preprint arXiv:2211.07377*, 2022.

[38] P. Tabuada and B. Gharesifard, "Universal Approximation Power of Deep Residual Neural Networks via Nonlinear Control Theory," in *Proceedings of the International Conference on Learning Representations*, 2021.

[39] K. Elamvazhuthi, B. Gharesifard, A. L. Bertozzi, and S. Osher, "Neural ODE Control for Trajectory Approximation of Continuity Equation," *IEEE Control Systems Letters*, vol. 6, pp. 3152–3157, 2022.

[40] M. Marchi, J. Bunton, B. Gharesifard, and P. Tabuada, "Safety and Stability Guarantees for Control Loops with Deep Learning Perception," *IEEE Control Systems Letters*, vol. 6, pp. 1286–1291, 2022.

[41] F. Abdollahi, H. A. Talebi, and R. V. Patel, "A stable neural network observer with application to flexible-joint manipulators," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 118–129, 2006.

[42] P. Bernard, "Observer Design for Nonlinear Systems," in *Lecture Notes in Control and Information Sciences*, vol. 479, Springer International Publishing, 2019.

[43] P. Bernard, V. Andrieu, and D. Astolfi, "Observer Design for Continuous-Time Dynamical Systems," *Annual Reviews in Control*, 2022.

[44] A. Tornambé, "High-gain observers for non-linear systems," *International Journal of Systems Science*, vol. 23, no. 9, pp. 1475–1489, 1992.

[45] A. Gelb, *Applied Optimal Estimation*. The MIT Press, 1974.

[46] N. Kazantzis and C. Kravaris, "Nonlinear observer design using lyapunov's auxiliary theorem," *Systems & Control Letters*, vol. 34, no. 5, pp. 241–247, 1998.

[47] V. Andrieu and L. Praly, "On the existence of a Kazantzis-Kravaris/Luenberger observer," *SIAM Journal on Control and Optimization*, vol. 45, no. 2, pp. 422–456, 2006.

[48] B. Schölkopf and A. J. Smola, *Learning with Kernels - Support Vector Machines, Regularization, Optimization, and Beyond* (Adaptive Computation and Machine Learning). The MIT Press, 2018.

[49] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Joint state and dynamics estimation with high-gain observers and Gaussian process models," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1627–1632, 2021.

[50] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Recognition Models to Learn Dynamics from Partial Observations with Neural ODEs," *Transactions on Machine Learning Research*, 2023.

[51] P.-F. Massiani, M. Buisson-Fenet, F. Solowjow, F. Di Meglio, and S. Trimpe, "Data-Driven Observability Analysis for Nonlinear Stochastic Systems," *Preprint arXiv:2302.11979*, 2023.

[52] M. Buisson-Fenet, L. Bahr, and F. Di Meglio, "Towards gain tuning for numerical KKL observers," *Preprint arXiv:2204.00318*, 2022.

[53] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[54] M. A. Álvarez, D. Luengo, M. K. Titsias, and N. D. Lawrence, "Variational inducing kernels for sparse convolved multiple output Gaussian processes," *Preprint arXiv:0912.3268*, pp. 1–22, 2009.

[55] I. Steinwart and A. Christmann, *Support Vector Machines*. Springer, New York, NY, 2008.

[56] D. K. Duvenaud, "Automatic model construction with Gaussian processes," Ph.D. dissertation, University of Cambridge, Pembroke College, 2014.

[57] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[58] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational Inference: A Review for Statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[59] M. K. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, vol. 5, Clearwater Beach, Florida, 2009, pp. 567–574.

[60] J. Hensman, A. G. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," in *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, vol. 38, San Diego, CA, USA, 2015, pp. 351–360.

[61] M. F. Huber, "Recursive Gaussian process regression," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3362–3366, 2013.

[62] M. Liu, G. Chowdhary, B. Castra Da Silva, S. Y. Liu, and J. P. How, "Gaussian Processes for Learning and Control: A Tutorial with Examples," *IEEE Control Systems Magazine*, vol. 38, no. 5, pp. 53–86, 2018.

[63] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," *Proceedings of the 57th IEEE Conference on Decision and Control*, pp. 6059–6066, 2018.

[64] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," *Proceedings of the 28th International Conference on Machine Learning*, pp. 465–472, 2011.

[65] A. Doerr, C. Daniel, D. Nguyen-Tuong, *et al.*, "Optimizing long-term predictions for model-based policy search," *Proceedings of the 1st Conference on Robot Learning*, vol. 78, pp. 227–238, 2017.

[66] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *Proceedings of the International Conference on Machine Learning*, pp. 1015–1022, 2010.

[67] A. Lederer, J. Umlauft, and S. Hirche, "Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control," in *Advances in Neural Information Processing Systems*, 2019, pp. 657–667.

[68] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively Learning Gaussian Process Dynamics," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 120, 2020, pp. 5–15.

[69] A. Capone, G. Noske, J. Umlauft, T. Beckers, A. Lederer, and S. Hirche, "Localized active learning of Gaussian process state space models," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 120, 2020, pp. 160–169.

[70] R. Turner, M. P. Deisenroth, and C. E. Rasmussen, "State-Space Inference and Learning with Gaussian Processes," *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 868–875, 2010.

[71] J. T. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth, "Efficiently sampling functions from Gaussian process posteriors," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 10 223–10 233.

[72] A. R. Geist and S. Trimpe, "Learning Constrained Dynamics with Gauss Principle adhering Gaussian Processes," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020.

[73] L. Rath, A. R. Geist, and S. Trimpe, "Using Physics Knowledge for Learning Rigid-body Forward Dynamics with Gaussian Process Force Priors," in *Proceedings of the 5th Conference on Robot Learning*, 2021, pp. 101–111.

[74] A. R. Geist and S. Trimpe, "Structured learning of rigid-body dynamics : A survey and unified view," *GAMM-Mitteilungen*, no. 44:e202100009, 2021.

[75] A. Besginow and M. Lange-Hegermann, "Constraining Gaussian Processes to Systems of Linear Ordinary Differential Equations," in *Advances in Neural Information Processing Systems*, 2022.

[76] K. Ensinger, F. Solowjow, M. Tiemann, and S. Trimpe, "Structure-preserving Gaussian Process Dynamics," *Preprint arXiv:2102.01606*, 2022.

[77] J. Brüdigam, M. Schuck, A. Capone, S. Sosnowski, and S. Hirche, "Structure-Preserving Learning Using Gaussian Processes and Variational Integrators," in *Proceedings of the 4th Conference on Learning for Dynamics and Control*, vol. 168, 2022.

[78] J. Umlauft and S. Hirche, "Learning stochastically stable Gaussian process state–space models," *IFAC Journal of Systems and Control*, vol. 12, 2020.

[79] A. Capone and S. Hirche, "Interval observers for a class of nonlinear systems using Gaussian process models," *Proceedings of the European Control Conference*, pp. 1350–1355, 2019.

[80] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational Gaussian process state-space models," vol. 4, 2014, pp. 3680–3688.

[81] S. Eleftheriadis, T. F. Nicholson, M. P. Deisenroth, and J. Hensman, "Identification of Gaussian process state space models," in *Advances in Neural Information Processing Systems*, 2017, pp. 5310–5320.

[82] A. Doerr, C. Daniel, M. Schiegg, *et al.*, "Probabilistic recurrent state-space models," *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., pp. 1280–1289, 2018.

[83] J. Lindinger, B. Rakitsch, and C. Lippert, *Laplace Approximated Gaussian Process State-Space Models*, 2022.

[84] S. Melchior, F. Berkenkamp, S. Curi, and A. Krause, "Structured Variational Inference in Unstable Gaussian Process State Space Models," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 120, 2020, pp. 147–157.

[85] A. D. Ialongo, M. Van Der Wilk, and C. E. Rasmussen, "Closed-form Inference and Prediction in Gaussian Process State-Space Models," in *Advances in Neural Information Processing Systems*, 2017.

[86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[87] S. Skansi, *Introduction to Deep Learning - From Logical Calculus to Artifical Intelligence*. Springer, 2018.

[88] G. B. Orr and K.-R. Müller, *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science). Springer-Verlag Berlin Heidelberg, 1998.

[89] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k2)$," *Soviet Mathematics Doklady*, vol. 27, 372—376, 1983.

[90] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[91] A. Güne¸, G. Baydin, B. A. Pearlmutter, and J. M. Siskind, "Automatic Differentiation in Machine Learning: a Survey," *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.

[92] M. Ekman, *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, NLP, and Transformers using TensorFlow*. Addison-Wesley Professional.

[93] S. L. Brunton, J. L. Proctor, J. N. Kutz, and W. Bialek, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 15, pp. 3932–3937, 2016.

[94] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[95] C. Scharzenberger and J. Hays, "Learning To Estimate Regions Of Attraction Of Autonomous Dynamical Systems Using Physics-Informed Neural Networks," *Preprint arXiv:2111.09930*, 2021.

[96] F. Masi and I. Stefanou, "Multiscale modeling of inelastic materials with Thermodynamics-based Artificial Neural Networks (TANN)," *Computer Methods in Applied Mechanics and Engineering*, vol. 398, 2022.

[97] M. Lutter, C. Ritter, and J. Peters, "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning," in *Proceedings of the International Conference on Learning Representations*, 2019.

[98] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian Neural Networks," in *Advances in Neural Information Processing Systems*, 2019.

[99] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian Neural Networks," in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[100] Y. D. Zhong, B. Dey, and A. Chakraborty, "Benchmarking Energy-Conserving Neural Networks for Learning Dynamics from Data," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, vol. 144, 2021, pp. 1218–1229.

[101] A. Sosanya and S. Greydanus, "Dissipative Hamiltonian Neural Networks: Learning Dissipative and Conservative Dynamics Separately," *Preprint arXiv: 2201.10085*, 2022.

[102] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[103] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," in *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.

[104] F. Masi and I. Stefanou, "Evolution TANN and the discovery of the internal variables and evolution equations in solid mechanics," *Preprint arXiv:2209.13269*, 2022.

[105] E. de Brouwer, J. Simm, A. Arany, and Y. Moreau, "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series," in *Advances in Neural Information Processing Systems*, 2019.

[106] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends in Machine Learning*, 2017.

[107] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer New York, NY, 2004.

[108] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[109] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. Sriperumbudur, "Gaussian Processes and Kernel Methods : A Review on Connections and Equivalences," *Preprint arXiv:1807.02582*, 2018.

[110] B. Schölkopf, R. Herbrich, and A. J. Smola, "A Generalized Representer Theorem," in *Proceedings of the 14th Annual Conference on Computational Learning Theory*, Springer Berlin Heidelberg, 2001, pp. 416–426.

[111] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, pp. 723–773, 2012.

[112] F. Solowjow, D. Baumann, C. Fiedler, A. Jocham, T. Seel, and S. Trimpe, "A Kernel Two-sample Test for Dynamical Systems," *Preprint arXiv:2004.11098*, 2021.

[113] E. D. Sontag, *Mathematical Control Theory - Deterministic Finite Dimensional Systems*, 2nd ed. Springer-Verlag New York, 1998.

[114] P. Bernard and V. Andrieu, "Luenberger Observers for Nonautonomous Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 270–281, 2019.

[115] M. Spirito, P. Bernard, and L. Marconi, "On the existence of robust functional KKL observers," in *Proceedings of the American Control Conference*, 2022.

[116] L da Costa Ramos, F Di Meglio, L. F. Figuiera da Silva, P Bernard, and V Morgenthaler, "Numerical design of Luenberger observers for nonlinear systems," in *Proceedings of the 59th Conference on Decision and Control*, 2020, pp. 5435–5442.

[117] J. Peralez and M. Nadri, "Deep Learning-based Luenberger observer design for discrete-time nonlinear systems," in *Proceedings of the IEEE Conference on Decision and Control*, IEEE, 2021, pp. 4370–4375.

[118] M. U. B. Niazi, J. Cao, X. Sun, A. Das, and K. H. Johansson, "Learning-based Design of Luenberger Observers for Autonomous Nonlinear Systems," *Preprint arXiv:2210.01476*, 2022.

[119] A. M. Dabroom and H. K. Khalil, "Discrete-time implementation of high-gain observers for numerical differentiation," *International Journal of Control*, vol. 72, no. 17, pp. 1523–1537, 1999.

[120] D. Astolfi, L. Marconi, L. Praly, and A. R. Teel, "Low-power peaking-free high-gain observers," *Automatica*, vol. 98, pp. 169–179, 2018.

[121] A. Chakrabarty, A. Zemouche, R. Rajamani, and M. Benosman, "Robust Data-Driven Neuro-Adaptive Observers with Lipschitz Activation Functions," in *Proceedings of the 58th IEEE Conference on Decision and Control*, 2019, pp. 2862–2867.

[122] A. Chakrabarty and M. Benosman, "Safe learning-based observers for unknown nonlinear systems using Bayesian optimization," *Automatica*, vol. 133, 2021.

[123] M. Bin and L. Marconi, "Model Identification and Adaptive State Observation for a Class of Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 5621 –5636, 2020.

[124] P. M. Cohn, *Further Algebra and Applications*. Springer-Verlag London Ltd., 2003.

[125] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, "Sliding Mode Control and Observation," in *Control Engineering*, Birkhäuser, New York, 2016.

[126] I. Thorson and D. Caldwell, "A nonlinear series elastic actuator for highly dynamic motions," *IEEE International Conference on Intelligent Robots and Systems*, pp. 390–394, 2011.

[127] D. Auroux and J. Blum, "Back and forth nudging algorithm for data assimilation problems," *Comptes Rendus l'Académie des Sciences, Série I, Mathématique*, vol. 340, no. 12, pp. 873–878, 2005.

[128] D. Auroux and J. Blum, "A nudging-based data assimilation method: The Back and Forth Nudging (BFN) algorithm," *Nonlinear Processes in Geophysics*, vol. 15, no. 2, pp. 305–319, 2008.

[129] E. Bullinger and F. Allgöwer, "An Adaptive High-Gain Observer," in *Proceedings of the IEEE Conference on Decision and Control*, 1997, pp. 4348–4353.

[130] R. G. Sanfelice and L. Praly, "On the performance of high-gain observers with gain adaptation under measurement noise," *Automatica*, vol. 47, no. 10, pp. 2165–2176, 2011.

[131] N. Boizot, E. Busvelle, and J. P. Gauthier, "An adaptive high-gain observer for nonlinear systems," *Automatica*, vol. 46, no. 9, pp. 1483–1488, 2010.

[132] S. Särkkä, "Bayesian Filtering and Smoothing," in *Institute of Mathematical Statistics Textbooks*, Cambridge University Press, 2010, pp. 1–232.

[133] K. Esfandiari and M. Shakarami, "Bank of High-Gain Observers in Output Feedback Control: Robustness Analysis Against Measurement Noise," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2019.

[134] Y. Rubanova, R. T. Chen, and D. Duvenaud, "Latent ODEs for irregularly-sampled time series," in *Advances in Neural Information Processing Systems*, 2019.

[135] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Neural Networks with Physics-Informed Architectures and Constraints for Dynamical Systems Modeling," in *Proceedings of the 4th Conference on Learning for Dynamics and Control*, vol. 168, 2022, pp. 263–277.

[136] O. Nelles, *Nonlinear System Identification*. Springer, Berlin, Heidelberg, 2001.

[137] L Praly, L Marconi, and A Isidori, "A new observer for an unknown harmonic oscillator," in *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, 2006, pp. 996–1001.

[138] Y. Yin, V. Le Guen, J. Dona, *et al.*, "Augmenting physical models with deep networks for complex dynamics forecasting," in *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[139] V. Mehta, I. Char, W. Neiswanger, *et al.*, "Neural Dynamical Systems," in *International Conference on Learning Representations - Integration of Deep Neural Models and Differential Equations Workshop*, 2020.

[140] J. L. Wu, C. Michelén-Ströfer, and H. Xiao, "Physics-informed covariance kernel for model-form uncertainty quantification with application to turbulent flows," *Computers and Fluids*, vol. 193, 2019.

[141] Ç. Yildiz, M. Heinonen, and H. Lähdesmäki, "ODE2VAE: Deep generative second order ODEs with Bayesian neural networks," in *Advances in Neural Information Processing Systems*, 2019.

[142] A. Norcliffe, C. Bodnar, B. Day, J. Moss, and P. Liò, "Neural ODE Processes," in *Proceedings of the International Conference on Learning Representations*, 2021.

[143] T. Doyeon, K. Thomas, Z. Luo, J. W. Pillow, and C. D. Brody, "Inferring Latent Dynamics Underlying Neural Population Activity via Neural Differential Equations," in *Proceedings of the 38th International Conference on Machine Learning*, 2021, pp. 5551–5561.

[144] F. M. Abushaqra, H. Xue, Y. Ren, and F. D. Salim, "CrossPyramid: Neural Ordinary Differential Equations Architecture for Partially-observed Time-series," *Preprint arXiv:2212.03560*, 2022.

[145] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, 2018.

[146] P. Bevanda, M. Beier, S. Kerz, A. Lederer, S. Sosnowski, and S. Hirche, "KoopmanizingFlows: Diffeomorphically Learning Stable Koopman Operators," *Preprint arXiv:2112.04085*, 2021.

[147] J. Hwang, J. Choi, H. Choi, K. Lee, D. Lee, and N. Park, "Climate Modeling with Neural Diffusion Equations," *Preprint arXiv:2111.06011*, 2021.

[148] M. Alexe and A. Sandu, "Forward and adjoint sensitivity analysis with continuous explicit Runge-Kutta schemes," *Applied Mathematics and Computation*, vol. 208, no. 2, pp. 328–346, 2009.

[149] A. Sandu, D. N. Daescu, and G. R. Carmichael, "Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: Part I - Theory and software tools," *Atmospheric Environment*, vol. 37, no. 36, pp. 5083–5096, 2003.

[150] H. Aliee, F. J. Theis, and N. Kilbertus, "Beyond Predictions in Neural ODEs: Identification and Interventions," *Preprint arXiv:2106.12430*, 2021.

[151] A. H. Ribeiro, K. Tiels, J. Umenberger, T. B. Schön, and L. A. Aguirre, "On the smoothness of nonlinear system identification," *Automatica*, vol. 121, 2020.

[152] D. Georges, "Machine Learning for Receding Horizon Observer Design : Application to Traffic Density Estimation," in *Proceedings of the 1st Virtual IFAC World Congress*, Berlin, Germany, 2020.

[153] E. Dupont, A. Doucet, and Y. W. Teh, "Augmented Neural ODEs," in *Advances in Neural Information Processing Systems*, 2019.

[154] M. Chalvidal, M. Ricci, R. VanRullen, and T. Serre, "Go with the Flow: Adaptive Control for Neural ODEs," in *Proceedings of the International Conference on Learning Representations*, 2021.

[155] A. Norcliffe, C. Bodnar, B. Day, N. Simidjievski, and P. Liò, "On second order behaviour in augmented neural ODEs," in *The Symbiosis of Deep Learning and Differential Equations Workshop (NeurIPS 2021)*, 2021.

[156] I. Ayed, E. D. Bezenac, A. Pajot, and P. Gallinari, "Learning the Spatio-Temporal Dynamics of Physical Processes from Partial Observations," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 3232–3236.

[157] A. Schlaginhaufen, P. Wenk, A. Krause, and F. Dörfler, "Learning Stable Deep Dynamics Models for Partially Observed or Delayed Dynamical Systems," in *Advances in Neural Information Processing Systems*, 2021.

[158] T. Kailath, *Linear Systems*. Englewood Cliffs, New Jersey: Prentice Hall PTR, 1980.

[159] S. Janny, V. Andrieu, M. Nadri, and C. Wolf, "Deep KKL: Data-driven Output Prediction for Non-Linear Systems," in *Proceedings of the IEEE Conference on Decision and Control*, 2021.

[160] S. Janny, Q. Possamaï, L. Bako, M. Nadri, and C. Wolf, "Learning Reduced Nonlinear State-Space Models : an Output-Error Based Canonical Approach," in *Proceedings of the 61st IEEE Conference on Decision and Control*, 2022.

[161] A. Gu, K. Goel, and C. Ré, "Efficiently Modeling Long Sequences with Structured State Spaces," in *Proceedings of the International Conference on Learning Representations*, 2022.

[162] A. Gu, I. Johnson, K. Goel, *et al.*, "Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers," in *Advances in Neural Information Processing Systems*, 2021, pp. 572–585.

[163] J. Eichelsdörfer, S. Kaltenbach, and P.-S. Koutsourelakis, "Physics-enhanced Neural Networks in the Small Data Regime," *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021)*, 2021.

[164] G. Manek and J. Zico Kolter, "Learning stable deep dynamics models," in *Advances in Neural Information Processing Systems*, 2019.

[165] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control," in *International Conference on Learning Representations*, 2020.

[166] S. Massaroli, M. Poli, M. Bin, J. Park, A. Yamashita, and H. Asama, "Stable Neural Flows," *Preprint arXiv:2003.08063*, 2020.

[167] M. Zakwan, L. Di Natale, B. Svetozarevic, P. Heer, C. N. Jones, and G. Ferrari Trecate, "Physically Consistent Neural ODEs for Learning Multi-Physics Systems," *Preprint arXiv:2211.06130*, 2022.

[168] K. L. Course, T. W. Evans, and P. B. Nair, "Weak Form Generalized Hamiltonian Learning," in *Advances in Neural Information Processing Systems*, 2020.

[169] M. Zhu, J. Moss, and P. Lio, "Modular Neural Ordinary Differential Equations," *Preprint arXiv:2109.07359v2*, 2019.

[170] B. Winkel. "2017 - Gustafson, G. B. - Differential Equations Course Materials." (2017), [Online]. Available: https://www.simiode.org/resources/3892.

[171] D. Karlsson and O. Svanström, "Modelling Dynamical Systems Using Neural Ordinary Differential Equations," Master's thesis in Complex Adaptive Systems, Chalmers University of Technology, Department of Physics, Tech. Rep., 2019.

[172] Q. Clairon and A. Samson, "Optimal control for estimation in partially observed elliptic and hypoelliptic linear stochastic differential equations," *Statistical Inference for Stochastic Processes*, vol. 23, pp. 105–127, 2020.

[173] M. Vigne, "Estimation and Control of the Deformations of an Exoskeleton using Inertial Sensors," Ph.D. dissertation, Mines ParisTech - Université PSL, 2021.

[174] A. J. Krener, "The Convergence of the Extended Kalman Filter," in *Directions in Mathematical Systems Theory and Optimization - Lecture Notes in Control and Information Sciences*, vol. 286, Springer-Verlag Berlin Heidelberg, 2003, pp. 173–182.

[175] A. Dulny, A. Hotho, and A. Krause, "NeuralPDE: Modelling Dynamical Systems from Data," *Preprint arXiv:2111.07671*, 2021.

[176] X. Xu, A. Hasan, K. Elkhalil, J. Ding, and V. Tarokh, "Characteristic Neural Ordinary Differential Equations," *Preprint arXiv:2111.13207*, 2021.

[177] C. Saunders, A. Gammerman, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables," in *Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 515–521.

[178] Z. Szabó, Z. Szabó, H. S.-c. Kernel, and M. Advances, "Hard Shape-Constrained Kernel Machines," in *Advances in Neural Information Processing Systems*, 2020.

[179] P.-C. Aubin-Frankowski and Z. Szabo, "Handling Hard Affine SDP Shape Constraints in RKHSs," no. 1955, 2021.

[180] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *Advances in Neural Information Processing Systems*, 2020.

[181] P. Hegde, Ç. Yıldız, H. Lähdesmäki, S. Kaski, and M. Heinonen, "Variational multiple shooting for Bayesian ODEs with Gaussian processes," in *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, 2022, pp. 790–799.

[182] D. Simon, *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[183] R. Li, Q. Zhang, J. Zhang, and T. Chu, "Distributional observability of probabilistic Boolean networks," *Systems & Control Letters*, 2021.

[184] E. Fornasini and M. E. Valcher, "Observability and Reconstructibility of Probabilistic Boolean Networks," *IEEE Control Systems Letters*, vol. 4, pp. 319–324, 2020.

[185] R. Zhou, Y. Guo, and W. Gui, "Set reachability and observability of probabilistic boolean networks," *Automatica*, 2019.

[186] I. Hwang, H. Balakrishnan, and C. Tomlin, "Observability criteria and estimator design for stochastic linear hybrid systems," in *Proceedings of the European Control Conference*, 2003.

[187] W. Zhang and B.-S. Chen, "On stabilizability and exact observability of stochastic systems with their applications," *Automatica*, vol. 40, pp. 87–94, 2004.

[188] H. Sun and M. Li, "Exact observability/exact detectability and spectrum assignment of stochastic systems," in *World Congress on Intelligent Control and Automation*, 2010, pp. 3786–3790.

[189] N. Powel and K. A. Morgansen, "Empirical Observability Gramian for Stochastic Observability of Nonlinear Systems," 2020, Preprint arXiv:2006.07451.

[190] Y. Sunahara, S. Aihara, and M. Shiraiwa, "The stochastic observability for noisy non-linear stochastic systems," *International Journal of Control*, pp. 461–480, 1975.

[191] V. Dragan and T. Morozan, "Stochastic observability and applications," *IMA Journal of Mathematical Control and Information*, vol. 21, pp. 323–344, 2004.

[192] N. D. Powel and K. A. Morgansen, "Empirical observability Gramian rank condition for weak observability of nonlinear systems with control," in *Proceedings of the 54th IEEE Conference on Decision and Control*, IEEE, 2015, pp. 6342–6348.

[193] C. Himpe. "Emgr—The Empirical Gramian Framework." (2018), [Online]. Available: https://www.mdpi.com/1999-4893/11/7/91 (visited on 05/31/2022).

[194] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. Oxford University Press, 1999.

[195] A. R. Liu and R. R. Bitmead, "Stochastic observability in network state estimation and control," *Automatica*, vol. 47, pp. 65–78, 2011.

[196] Y. Subasi and M. Demirekler, "Quantitative measure of observability for linear stochastic systems," *Automatica*, vol. 50, pp. 1669–1674, 2014.

[197] L. Song, K. Fukumizu, and A. Gretton, "Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 98–111, 2013.

[198] K. Fukumizu, L. Song, and A. Gretton, "Kernel Bayes' rule: Bayesian inference with positive definite kernels," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3753–3783, 2013.

[199] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. 49–57, 2006.

[200] B. K. Sriperumbudur, K. Fukumizu, and G. R. G. Lanckriet, "Universality, Characteristic Kernels and RKHS Embedding of Measures," *Journal of Machine Learning Research*, 2011.

[201] B. Efron, "Bootstrap methods: Another look at the jackknife," in *Breakthroughs in Statistics: Methodology and Distribution*. Springer New York, 1992, pp. 569–593.

[202] M. J. Aburn and Y. Ram, *Numerical integration of stochastic differential equations (SDEs)*, Accessed Sept. 7th, 2022, 2022.

[203] D. Astolfi, R. Postoyan, and D. Nesic, "Uniting local and global observers for the state estimation of nonlinear continuous-time systems," in *Proceedings of the 56th IEEE Conference on Decision and Control*, 2018, pp. 3039–3044.

[204] Quanser, *Quanser courseware and resources*, https://www.quanser.com/products/qube-servo-2/, 2022. (visited on 07/12/2022).

[205] D. Luenberger, "Observers for multivariable systems," *IEEE Transactions on Automatic Control*, vol. 11, no. 2, pp. 190–197, 1966.

[206] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, pp. 95–108, 1 1961.

[207] G. Bornard and H. Hammouri, "A high gain observer for a class of uniformly observable systems," in *Proceedings of the 30th IEEE Conference on Decision and Control*, 1991, pp. 1494 –1496.

[208] H. K. Khalil and L. Praly, "High-gain observers in nonlinear feedback control," *International Journal of Robust and Nonlinear Control*, vol. 24, pp. 993–1015, 2014.

[209] A. J. Krener, "The convergence of the Extended Kalman Filter," in *Directions in mathematical systems theory and optimization*, Springer-Verlag Berlin Heidelberg, 2003, pp. 173–182.

[210] M. Maggiore and K. M. Passino, "A separation principle for a class of non-uco systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1122–1133, 2003.

[211] N. Henwood, "Estimation en ligne de paramètres de machines electriques pour véhicule en vue d'un suivi de la température de ses composants," Ph.D. dissertation, Mines ParisTech, 2014.

[212] H. Toivonen, "Signal and system norms," *Lecture Notes for the Course "Advanced Control Methods"*, 2010.

[213] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," *Advances in Neural Information Processing Systems*, pp. 3835–3844, 2018.

[214] A. McGlinchey and O. Mason, "Bounding the l2 sensitivity for positive linear observers," in *Proceedings of the European Control Conference*, 2018, pp. 1214–1219.

[215] S. Nikiin, "Sensitivity of Luenberger Observers, e-Oberservability and Uncertainty Relations," Universität Kaiserslautern - Fachbereich Mathematik, Tech. Rep., 1992.

[216] J. W. Xu, D. Erdogmus, and J. C. Principe, "Minimum error entropy Luenberger observer," in *Proceedings of the American Control Conference*, IEEE, 2005, pp. 1923–1928.

[217] N. Rouche, P. Habets, and M. Laloy, *Stability theory by Liapunov's direct method*. Springer New York, NY, 1977, vol. 22.

[218] L. Marconi and L. Praly, "Uniform practical nonlinear output regulation," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1184–1202, 2008.

[219] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

[220] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning," in *Third workshop on Bayesian Deep Learning (NeurIPS 2018)*, 2018.

[221] C. Doersch, "Tutorial on Variational Autoencoders," *Preprint arXiv:1606.05908*, pp. 1–23, 2016.

[222] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *International Conference on Learning Representations*, 2013.

[223] K. Miao and K. Gatsis, "Learning Robust State Observers using Neural ODEs (longer version)," *Preprint arXiv:2212.00866*, 2022.

[224] K. Pons and M. Ersoy, "Adaptive mesh refinement method . Part 1 : Automatic thresholding based on a distribution function," *HAL preprint hal-01330679*, 2016.

[225] K. K. Vu, C. Ambrosio, Y. Hamadi, and L. Liberti, "Surrogate-based methods for black-box optimization," *International Transactions in Operational Research*, vol. 24, no. 3, pp. 393–424, 2017.

RÉSUMÉ

La modélisation et la simulation numérique de processus complexes sont aujourd'hui des éléments essentiels du développement industriel. De par la récente augmentation des capacités de génération, de collecte et de traitement des données, les méthodes basées sur l'apprentissage apparaissent aujourd'hui comme un complément prometteur à la modélisation physique. Unir ces deux points de vue permettrait notamment de créer des jumeaux digitaux : des reproductions numériques exactes d'objets physiques combinant un modèle de simulation haute-fidélité et des données expérimentales recueillies sur le système réel. Cependant, les données disponibles sur les plateformes physiques sont généralement bruitées et tous les états ne peuvent pas être mesurés. Notre objectif est d'extraire des informations de ces données expérimentales sur le modèle sous forme d'état sous-jacent.

D'une part, si le système doit être identifié à partir des données, cette information peut prendre la forme d'un modèle dynamique. En raison de la nature partielle des observations, il est nécessaire d'estimer conjointement l'état sous-jacent et sa dynamique. Nous exploitons des concepts d'estimation d'état dans des méthodes modernes d'apprentissage de la dynamique pour réaliser l'identification du système à partir de ces observations, d'abord pour une forme spécifique de systèmes permettant des garanties théoriques, puis dans un cadre plus général. D'autre part, ces informations peuvent porter sur l'estimation d'état elle-même. Nous proposons d'analyser l'observabilité à partir des données de sortie en utilisant des outils statistiques. Nous tirons ensuite parti des techniques modernes d'apprentissage pour construire des observateurs numériques pour les systèmes non linéaires.

MOTS CLÉS

Identification de systèmes, Apprentissage automatique, Données expérimentales, Design d'observateurs, Jumeaux numériques

ABSTRACT

Numerical simulation and modeling of complex processes is a critical part of industrial development. With the recent increase in data generation, collection and processing capabilities, learning-based methods appear as a promising addition to physics-based modeling. Uniting both views is an appealing prospect, e.g., to create digital twins: exact numerical replicas of physical objects combining a high-fidelity simulation model with experimental data gathered on the real system. However, the data available from physical platforms may be noisy and not cover all state variables. Our aim is to extract information from this experimental data about the underlying state-space model that explains it.

On the one hand, if the system needs to be identified from data, this information can take the form of a dynamics model. Due to the partial nature of the observations, it is necessary to jointly estimate the underlying latent state and its dynamics. We leverage concepts from state estimation in modern dynamics learning methods to achieve system identification from these observations, first for a specific form of systems allowing for theoretical guarantees, then in a more general setting.

On the other hand, this information can relate to state estimation itself. We draw on machine learning techniques to enable system theoretic analysis from the measurements. We then propose to analyze observability from output data and to leverage modern machine learning techniques to build numerical observers for general nonlinear systems.

KEYWORDS

System identification, Machine learning, Experimental data, Observer design, Digital twins