PHILIPPE WENK

LEARNING TIME-CONTINUOUS DYNAMICS MODELS WITH GAUSSIAN-PROCESS-BASED GRADIENT MATCHING

Diss. ETH No. 28730

DISS. ETH NO. 28730

LEARNING TIME-CONTINUOUS DYNAMICS MODELS WITH GAUSSIAN-PROCESS-BASED GRADIENT MATCHING

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

PHILIPPE WENK

born on 25 December 1991
MSc RSC, ETH Zürich

accepted on the recommendation of

Dr. Andreas Krause, examiner
Dr. Maurizio Filippone, co-examiner
Dr. Florian Dörfler, co-examiner
Dr. Stefan Bauer, co-examiner

19.12.2022

Philippe Wenk: *Learning Time-Continuous Dynamics Models with Gaussian-Process-based Gradient Matching*, © 19.12.2022

DOI: 10.3929/ethz-b-000591579

ABSTRACT

Continuous-time dynamical systems form an important modeling class in many scientific disciplines. Thus, combining them with a solid uncertainty quantifying framework could enable important applications, including guidance for data selection or for safety considerations. Nevertheless, uncertainty quantification in continuous-time is notoriously difficult, both theoretically and computationally. A key issue here is the computational complexity and sometimes numerical instability of numerical integration.

In this thesis, we investigate alternatives to classical, numerical-integration-based approaches to probabilistic inference for such systems, namely gradient matching. After a general introduction to gradient matching schemes, we develop novel, Gaussian-process-based inference schemes that do not rely on numerical integration. We start in a theoretical setting, where we can guarantee that the underlying, data-generating process can be described by specific models. In particular, the first models we consider are deterministic, ordinary differential equations. Then, our inference algorithms are adapted and scaled up. Ultimately, they are deployed in the context of a data set containing real world measurements, in a case study where some of the underlying assumptions are demonstrably violated. The resulting algorithm is a compromise between pure gradient-matching-based and numerical-integration-based techniques. While we still avoid numerical integration during training, we observe that it can be helpful and necessary during prediction.

By building on insights developed in the deterministic case, we then conclude our work by studying the applicability of gradient matching to stochastic differential equations. By restricting our work to linear diffusion terms, we ultimately manage to develop a gradient matching algorithm for this model class, even though sample paths of stochastic differential equations are nowhere differentiable with probability 1.

ZUSAMMENFASSUNG

Zeitkontinuierliche dynamische Systeme sind eine wichtige Klasse von Modellen für viele wissenschaftliche Disziplinen. Ein solider Ansatz zur Quantifizierung von Unsicherheit in solchen Systemen könnte dadurch zu vielen interessanten Anwendungen führen, z.B. bei der Auswahl von zukünftigen Experimenten oder zur Zertifizierung von Sicherheitsüberlegungen. Dennoch ist die Quantifizierung von Unsicherheit in zeitkontinuierlichen Systemen sowohl theoretisch als auch rechnerisch sehr schwierig. Ein Kernproblem ist dabei der hohe Rechenaufwand assoziiert mit numerischer Integration, sowie dabei auftretende, numerische Probleme.

In dieser Arbeit untersuchen wir Alternativen zu klassischen, auf numerischer Integration basierenden Ansätzen zur probabilistischen Inferenz für solche Systeme, genannt Gradienten-Matching. Nach einer allgemeinen Einführung in Gradienten-Matching-Verfahren entwickeln wir neuartige, auf Gauss-Prozessen basierende Inferenzverfahren, die nicht auf numerischer Integration beruhen. Wir beginnen in einem theoretischen Experimentaufbau, in dem wir garantieren können, dass der zugrundeliegende, datenerzeugende Prozess durch bestimmte Modelle beschrieben werden kann. Die ersten Modelle, die wir betrachten, sind einfache, deterministische Differentialgleichungen. Aufbauend auf diese Betrachtungen erweitern wir dann unsere Inferenzalgorithmen, bis sie schliesslich im Kontext eines Datensatzes eingesetzt werden können, der reale Messungen enthält. Dies führt zu einer Fallstudie, in der einige der zugrundeliegenden Annahmen nachweislich verletzt werden. Der resultierende Algorithmus ist ein Kompromiss zwischen reinen Gradienten-Matching-Verfahren und Verfahren mit numerischer Integration. Während wir die numerische Integration beim Training noch vermeiden, stellen wir fest, dass sie bei der Vorhersage tatsächlich hilfreich und notwendig sein kann.

Aufbauend auf den Erkenntnissen, die im deterministischen Fall entwickelt wurden, schliessen wir unsere Arbeit ab, indem wir die Anwendbarkeit von Gradienten-Matching auf stochastische Differentialgleichungen untersuchen. Durch die Beschränkung unserer Arbeit auf lineare Diffusionsterme gelingt es uns, einen Gradienten-Matching Algorithmus für diese Modellklasse zu entwickeln, obwohl die einzelnen Pfade einer stochastischen Differentialgleichung mit Wahrscheinlichkeit 1 nirgends differenzierbar sind.

ACKNOWLEDGEMENTS

I would like to thank my two advisors, Andreas Krause and Stefan Bauer. Their continuous support, expertise and motivation to continuously explore new angles in more depth were a driving factor behind the success of this thesis. Thank you for all the invaluable discussions over the years. Along similar lines, I would like to thank Florian Dörfler for the many brilliant discussions and interesting inputs, especially during the second half of my Ph.D.. I would also like to thank Maurizio Filippone, who kindly agreed to serve on my Ph.D. committee.

During this thesis, I had the privilege to work with many very talented researchers. I am extremely grateful to all of my collaborators for the countless interesting discussion and invaluable inputs. In particular, I would like to thank Gabriele Abbati and Lenart Treven, with whom close collaborations resulted in joint first author publications. I am also grateful to had the opportunity to, to various extent, collaborate and discuss ideas with Bernhard Schölkopf, Michael Osborne, Joachim Buhmann, Alkis Gotovos, Andrea Ianelli and Nico Gorbach.

During my PhD, I had the great pleasure of supervising many interesting projects of talented master students. In particular, I would like to thank Edoardo Caldarelli, Andreas Schlaginhaufen and Emmanouil Angelis, with whom collaborations not only lead to many interesting inputs for my thesis, but whose excellent work also lead to independent publications. I am also grateful to had the opportunity and pleasure to work with Guanchun Tong, Julian Roth, Eshref Özdemir, Gautam Sridhar, Camilla Casamento Tumeo, Cédric Della Casa, Dimitar Dimitrov, Felix Schur, Robert Marks, Felix Crazzolara, Luis Wyss and Martin Xu.

Beyond publications, I would like to thank everyone at ETH that contributed both to my scientific development and the preservation of my sanity. Here, I would especially like to thank my colleagues Max Paulus, Andisheh Amrollahi, Alina Dubatovka, Djordje Miladinovic and Aytunc Sahin. And of course, I would like to thank all members of LAS group specifically and the machine learning group at ETH more generally. You provided me with a unique working experience and I am glad that our paths met.

I am also very grateful to Rita Klute and the administrative staff at ETH. Even before my journey officially started, during the pandemic and in

my final moments at ETH, your help and support was always immensely appreciated.

I am happy to acknowledge institutions that have supported the research in this dissertation, especially the Department of Computer Science at ETH Zürich, the Max Planck ETH Center for Learning Systems (CLS), and the ELLIS program.

Last but not least, I would like to thank my family and close friends. You supported me in countless moments and I am forever grateful. You know who you are.

CONTENTS

1 INTRODUCTION AND OVERVIEW 1
- 1.1 Problem Setting 1
  - 1.1.1 Ordinary Differential Equations 2
  - 1.1.2 Stochastic Differential Equations 2
- 1.2 Numerical Integration vs Collocation Methods 3
  - 1.2.1 Frequentist Parameter Inference 3
  - 1.2.2 Bayesian Parameter Inference 4
  - 1.2.3 Numerical Integration for SDEs 4
  - 1.2.4 Avoid Numerical Integration Via Gradient Matching 5
- 1.3 Outlook 6

2 THE BAYESIAN: FAST GAUSSIAN-PROCESS-BASED GRADIENT MATCHING 7
- 2.1 Introduction and Related Work 7
- 2.2 Problem Setting and Notation 8
- 2.3 GP-Based Gradient Matching - State Of The Art 9
  - 2.3.1 Calculating Derivatives: Basic Modeling Choices 10
  - 2.3.2 Merging the two Components Via the Product of Experts Heuristics 10
  - 2.3.3 State of the Art Inference 12
  - 2.3.4 Gradient Matching Without Product of Experts 13
- 2.4 Theory 13
  - 2.4.1 Analysis Of The Product Of Experts Approach 13
  - 2.4.2 Adapting The Original Graphical Model 14
  - 2.4.3 Inference In The New Model 16
- 2.5 Hyperparameters and Preprocessing 16
  - 2.5.1 Hyperparameter and kernel selection 17
  - 2.5.2 Accounting for Normalization and Standardization 18
- 2.6 FGPGM - Fast Gaussian Process Based Gradient Matching 20
- 2.7 Experiments 20
  - 2.7.1 Benchmark Tasks 21
  - 2.7.2 Evaluation 24
  - 2.7.3 Sampling vs Variational Inference - Lotka Volterra 25

2.7.4 Beyond Locally Linear Dynamics - Protein Transduction 26
2.7.5 Investigating Smoothness Bias - FitzHugh-Nagumo 28
3 THE FREQUENTIST: ODE-INFORMED REGRESSION 31
3.1 Problem Setting and Related Work 31
3.2 Methods 32
3.2.1 Notation 32
3.2.2 Generative Model 33
3.2.3 ODIN - ODE-Informed Regression 34
3.2.4 Derivative Observation Model 36
3.2.5 Remarks 36
3.3 Experiments 37
3.3.1 State and Parameter Inference 38
3.3.2 Model Selection 42
3.3.3 Linear Scaling in State Dimension 43
3.4 SLEIPNIR - Scaling to Bigger Datasets 43
3.4.1 Scaling standard GP Regression 44
3.4.2 Scaling GP regression with derivatives 45
3.4.3 Accurately Approximating Derivative Kernels 45
3.4.4 Accurately Approximating Posteriors for GP Regression with Observed Derivatives 51
3.4.5 Derivation 52
3.5 ODIN with SLEIPNIR - Risk Consistent Scaling 54
3.5.1 Derivation 55
3.6 Scaling Experiments 56
4 THE PRACTITIONER: DISTRIBUTIONAL GRADIENT MATCHING 59
4.1 Background 60
4.1.1 Data 60
4.1.2 Problem 61
4.1.3 Motivation 61
4.2 Distributional Gradient Matching 62
4.2.1 Regularization by Matching Distributions over Gradients 63
4.2.2 Smoothing jointly over Trajectories with Deep Gaussian Processes 63
4.2.3 Representing Uncertainty in the Dynamics Model via the Reparametrization Trick 66

4.2.4 Comparing Gradient Distributions via the Wasserstein Distance 67
4.2.5 Final Loss Function 68
4.3 Experiments 69
4.3.1 Setup 70
4.3.2 Metric 70
4.3.3 Effects of Overparametrization 70
4.3.4 Single Trajectory Benchmarks 71
4.3.5 Prediction speed 72
4.3.6 Multi-Trajectory Benchmarks 73
4.3.7 Ablation study 73
4.3.8 Computational Requirements 74
4.3.9 Scaling to many observations or trajectories 75
4.4 Case Study: USHCN 78
4.4.1 The data set 78
4.4.2 Preparing the data set 80
4.4.3 Challenge of the data set 82
4.4.4 Including dynamics helps with extrapolation 85
4.4.5 Comparisons 88
4.4.6 Conclusion 91
5 THE STOCHASTIC: ADVERSARIAL AND MMD-MINIMIZING REGRESSION 93
5.1 Introduction 93
5.1.1 Related Work 94
5.1.2 Our Work 95
5.2 Background 96
5.2.1 Deterministic ODE Case 96
5.2.2 Notation 97
5.3 Methods 98
5.3.1 Latent States Representation 98
5.3.2 Generative Model for Observations 99
5.3.3 Generative Model for Derivatives 101
5.3.4 Inference 102
5.3.5 Adversarial Sample-based Inference 103
5.3.6 Maximum Mean Discrepancy 104
5.4 Experiments 106
5.4.1 Setups 106
5.4.2 Evaluation 108
5.4.3 Locally Linear Systems 108

5.4.4 Non-Diagonal Diffusion 108
5.4.5 Dealing with Multi-Modality 109
6 SUMMARY 111
A APPENDIX 113
A.1 Appendix to FGPGM 113
A.1.1 Proof of theorem 1 113
A.1.2 Additional Plots 114
A.2 Appendix to ODIN 116
A.2.1 ODEs Provide Useful Information 116
A.2.2 Median Trajectories 116
A.2.3 Kernel Approximation Error Bounds - Proofs 117
A.2.4 Kernel Approximation Additional Plots 133
A.2.5 GP Regression with Derivatives 134
A.2.6 Additional Empirical Evaluation GPR 140
A.2.7 Risk Approximation Error Bounds 148
A.2.8 Experimental Setups 156
A.2.9 Additional Empirical Evaluation SLEIPNIR 160
A.2.10 tRMSE vs Features 160
A.2.11 Learning Curves 164
A.3 Appendix to DGM 169
A.3.1 Dataset description 170
A.3.2 Implementation details of DGM 178
A.3.3 Bayesian NODE training 182
A.3.4 Additional experiments 187
A.4 Appendix to ARES/MARS 204
A.4.1 Parameter Estimation Lorenz '63 204
A.4.2 Training Times 204
A.4.3 Densities for Ancestral Sampling of the SDE-Based Model 204
A.4.4 Calculating the GP Posterior for Data-Based Ancestral Sampling 206

BIBLIOGRAPHY 209

1

INTRODUCTION AND OVERVIEW

At the core of human consciousness lies the ability to model our environment and to reason about hypothetical actions counterfactually within these models. In science and engineering, such models are routinely created and deployed. In engineering in particular, the creation of such models is summarized in the field of system identification [Lju98]. System identification plays a crucial part in robotics and control, being a precursor to classical control algorithms like linear quadratic Gaussian (LQG) control [AM07] or more modern approaches like model predictive control (MPC) [GPM89]. Models for such applications can be grounded in physics, especially when considering gray box models [Soh98]. However, despite physics being fundamentally time-continuous, most active research studies time-discrete models. This mainly stems from the fact that time-continuous systems are more difficult to handle, both from a computational and from a statistical perspective [Zam+11]. This is especially true when we want to consider uncertainty estimation, which is important, e.g., for active / reinforcement learning algorithms [Set09] and important for reasoning about systems with safety constraints [GF15].

Thus, in this thesis, we aim to reduce both the statistical and computational obstacles that arise when working with uncertainty quantification in nonlinear, time-continuous systems. As a first step towards this ultimate goal, we exclusively focus on autonomous, time-independent systems and leave the study of other systems to future work.

1.1 PROBLEM SETTING

In particular, we will focus on learning models for time series data. A time series data set $\mathcal{D} = \{\mathcal{D}_d\}_{d=0}^{N_d}$ consists of N_d triplets $\mathcal{D}_i := \left(\boldsymbol{y}(t_d), \boldsymbol{x}(0)^{(d)}, t_d\right)$, consisting of an initial condition $\boldsymbol{x}(0)$, a time stamp t_d and an observation $\boldsymbol{y}(t_d)$. It is assumed that the observation $\boldsymbol{y}(t_d)$ is created by initializing an unknown dynamical system at time $t = 0$ with initial condition $\boldsymbol{x}^{(d)}(0)$. This system is then evolved until time t_d, at which a noisy observation of the system state $\boldsymbol{x}(t_d)$ is taken to observe $\boldsymbol{y}(t_d)$. Usually, the observation

noise is assumed to be i.i.d. Gaussian, while the structure of the underlying system will change from chapter to chapter.

1.1.1 *Ordinary Differential Equations*

In Chapters 2, 3 and 4, the underlying dynamical system is given by an ordinary differential equation (ODE) of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\theta}). \tag{1.1}$$

Here, $\boldsymbol{f}$ is a parametric function with parameters $\boldsymbol{\theta}$. The parametric form of $\boldsymbol{f}$ is assumed to be known, so the main goal of this learning task is to infer the unknown parameters $\boldsymbol{\theta}$. For all algorithms, it is required that $\boldsymbol{f}$ is globally uniform Lipschitz, so that unique solutions to the initial value problem exist. It should be noted that this assumption is the bare minimum necessary to have a meaningful problem definition. However, it is not particularly restrictive. Throughout Chapters 2, 3 and 4, we analyze traditional parametric models, as one would find for example in systems biology. However, in Chapter 4, we also demonstrate that within this framework, we can learn complicated neural dynamics models, where $\boldsymbol{f}$ is modelled as a neural network.

1.1.2 *Stochastic Differential Equations*

In Chapter 5, we leave the deterministic framework of ODEs and present an extension of the aforementioned algorithms to stochastic differential equations (SDEs). To describe SDEs, we exclusively use the Itô-form

$$d\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\theta})dt + \boldsymbol{G}(\boldsymbol{x}(t), \boldsymbol{\theta})d\boldsymbol{w}(t). \tag{1.2}$$

Again, $\boldsymbol{x}(t)$ denotes the time-dependent vector of states of the underlying dynamical system and $\boldsymbol{\theta}$ denotes the collection of all the parameters of the model. The dynamics are now given by a drift function $\boldsymbol{f}$, which represents the deterministic part of the dynamics, as well as a matrix-valued diffusion function $\boldsymbol{G}$, which modulates a stochastic Wiener-process $\boldsymbol{w}(t)$ of the same dimension as the state vector $\boldsymbol{x}$. As before, the parametric form of $\boldsymbol{f}$ and $\boldsymbol{G}$ is assumed to be known and the challenge of this task lies in inferring appropriate parameters $\boldsymbol{\theta}$.

1.2 NUMERICAL INTEGRATION VS COLLOCATION METHODS

Independently of the underlying system, inferring suitable parameters $\boldsymbol{\theta}$ requires some mathematical relationship quantifying how the choice of parameters influences the data generating process. To illustrate the problem, we first present examples for both frequentist and Bayesian parameter inference in the context of ODEs. We will then conclude by introducing the high-level idea of gradient matching, which will form the core of all algorithms presented in this thesis.

1.2.1 *Frequentist Parameter Inference*

Consider the ODE problem setting with dynamics as defined in Equation (1.1). Assume we want to find optimal system parameters $\boldsymbol{\theta}^*$ by minimizing the squared error of the observation fit, i.e.,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{d=0}^{N_d} ||\boldsymbol{y}(t_d) - \boldsymbol{x}(t_d)||^2, \tag{1.3}$$

where

$$\boldsymbol{x}(t_d) := \boldsymbol{x}(0)^{(d)} + \int_0^{t_d} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{\theta}) d\tau. \tag{1.4}$$

To solve the optimization problem of Equation (1.3), we would ideally have access to a closed-form expression of the map $\left(\boldsymbol{x}(0)^{(d)}, t_d, \boldsymbol{\theta}\right) \mapsto (\boldsymbol{x}(t_d), \nabla_{\boldsymbol{\theta}} \boldsymbol{x}(t_d))$. Such a map would allow us to efficiently evaluate both the loss and the gradient of the optimizaton problem. However, for most (nonlinear) interesting systems, such closed-form solutions do not exist. Thus, if gradient-based optimization should be deployed, one needs to find a way to differerntiate through the integral of Equation (1.4).

In the literature, this problem has been intensively studied [Bar74; Ben79], with the two most common approaches including calculating forward sensitivities or backward sensitivities via the adjoint method [Pon+62; Ma+21; Hou+12]. All of these classical approaches rely on propagating augmented states through the integral of Equation (1.4), which can come with its own numerical or computational challenges. However, as we will demonstrate in Chapters 4 and 5, such integration-based methods struggle in a stochastic or Bayesian setting.

1.2.2 *Bayesian Parameter Inference*

In the Bayesian parameter inference setting, the goal is not just to obtain one numerical value for $\boldsymbol{\theta}^*$, but to infer the whole posterior of the parameters $p(\boldsymbol{\theta} \mid \mathcal{D})$. For all but the simplest systems, calculating this posterior in closed-form is impossible. A more sensible approach consists of using Bayes' rule to obtain

$$p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta}). \tag{1.5}$$

Here, $p(\boldsymbol{\theta})$ is a prior, which is set before seeing training data. Its form is usually chosen such that the posterior is easy to evaluate. Nevertheless, especially if the parameters have some physical interpretation in the context of the underlying system, it can be used to incorporate prior knowledge by domain experts. Furthermore, $p(\mathcal{D} \mid \boldsymbol{\theta})$ can be calculated using Equation (1.4) in combination with a suitable observation model. Thus, the main challenge of evaluating the posterior lies in calculating the normalization constant of Equation (1.5). Again, there exists a plethora of approaches to deal with this problem, including MCMC-based methods [Ma+21; RC11] or variational approaches [BKM17]. However, all of them rely on tracing the influence of the parameters $\boldsymbol{\theta}$ on the states $\boldsymbol{x}$ through the integral of Equation (1.4). This can be a key weakness, both from a computational and from a numerical standpoint, as we will demonstrate in Chapter 4.

1.2.3 *Numerical Integration for SDEs*

Thus far, numerical integration of Equation (1.4) was shown to form a key component of many state-of-the-art ODE parameter inference algorithms. For SDEs, the equivalent step requires integrating Equation (1.2). Again, there exist only closed-form solutions for very few systems. However, numerical integration of Equation (1.2) is orders of magnitudes harder compared to the deterministic case, especially as $\boldsymbol{w}(t)$ is the realization of a stochastic process. Thus, even for fixed parameters $\boldsymbol{\theta}$, every path obtained via a suitable integration scheme will only represent one realization of Equation (1.2). Thus, a parameter inference scheme relying on numerical integration might require many sample paths just to judge the quality of one set of parameters. While we will discuss this problem in more detail in Chapter 5, this further demonstrates the need for alternative methods, not relying on numerical integration.

1.2.4 *Avoid Numerical Integration Via Gradient Matching*

Starting with the spline-based approach introduced by Varah [Var82], a family of gradient matching algorithms tries to avoid numerical integration completely.

CLASSICAL ALGORITHMS ARE PARAMETER-CENTERED To this end, we leave the parameter-centered methodology of classical approaches. As an example, consider parameter inference of ODEs. In a classical algorithm, we first choose a parameter set $\boldsymbol{\theta}$. Then, we create the corresponding path $\boldsymbol{x}(t)$ by numerically integrating Equation (1.4). This guarantees that we only consider candidate paths that satisfy the underlying dynamics. From this candidate pool, we then find the best path by choosing the one path that fits the observations the best, e.g., via the optimization given by Equation (1.3). In that sense, classical methods search the *space of paths agreeing with the dynamics*, as parametrized by the parameters $\boldsymbol{\theta}$, for a path that fits the observations well.

DYNAMICS AND STATES DO NOT NEED TO AGREE COMPLETELY At first, this seems like a reasonable proposition. Since we automatically enforce agreement with the dynamics, we only have to care about data fit. However, this viewpoint is overly optimistic. Since there are numerical approximations in every numerical integrator, the agreement with the dynamics will always be up to some tolerance. Thus, we try to guarantee agreement with the dynamics as much as possible, unconsciously prioritizing it over data fit.

GRADIENT MATCHING METHODS ARE PATH CENTERED Gradient matching algorithms take a different viewpoint. Here, the key entity are the paths $\boldsymbol{x}(t)$. As the classical approaches, we want to find a path that both fits the observations well, but that also satisfies agreement with the dynamics. Unlike the classical approaches though, we model the paths directly with some interpolator. This interpolator is chosen to be differentiable, so that at each time point, we can calculate $\boldsymbol{x}(t)$, i.e., the path itself, as well as its derivatives $\dot{\boldsymbol{x}}(t)$. Given $\boldsymbol{x}(t)$, quantifying the data fit is usually straight forward. Furthermore, pairs $(\boldsymbol{x}(t_i), \dot{\boldsymbol{x}}(t_i)$ of state and derivatives allow for calculating the agreement with the derivatives, by evaluating deviations between the interpolator gradients $\dot{\boldsymbol{x}}(t_i)$ and the corresponding dynamics derivatives $\boldsymbol{f}(\boldsymbol{x}(t_i), \boldsymbol{\theta})$. Clearly, the choice of t_i allow us to control the

desired accuracy of the evaluation of the agreement with the derivatives. Furthermore, since we can clearly distinguish data fit from dynamics agreement, they can be carefully balanced, potentially allowing for more effective or more amenable optimization landscapes.

1.3 OUTLOOK

In the literature, there exist many different versions of gradient matching. Given the high-level picture of the idea sketched in the previous section, gradient matching algorithms can differ in the problem setting they analyze, the choice of interpolation scheme or the inference method considered. In this thesis, we will exclusively focus on developing new and extending existing gradient matching schemes based on Gaussian process interpolators. Most of the thesis concentrates on parameter inference for ODEs. In Chapter 2, we discuss and extend the existing theory behind GP-based gradient matching in a Bayesian context. In Chapter 3, some key concepts and intuitions are analyzed in a frequentist context. Then, in Chapter 4, all of these insights are combined and extended to obtain an algorithm suitable for large, real world datasets with no or little expert knowledge available. Finally, we conclude the thesis by demonstrating how gradient matching methods can be deployed in the context of SDEs, i.e. how to match gradients in a setting where the gradients of the sample paths exist with probability 0. All chapters contain work that was created in collaboration with colleagues and that has been published before. Chapter 2 is based on [Wen+19], Chapter 3 on [Wen+20] and [Ang+20], the first part of Chapter 4 on [Tre+21] and Chapter 5 on [Abb+19].

2

THE BAYESIAN: FAST GAUSSIAN-PROCESS-BASED GRADIENT MATCHING

In this chapter, we study the problem of Bayesian parameter inference for parametric ordinary differential equations. This chapter is in large parts based on [Wen+19], which is work that profited from numerous discussions with and comments of its co-authors. Especially the theory and experiments section of this chapter correspond in large parts to the original theory and experiments section of [Wen+19], which was published as a conference paper. Furthermore, the presentation of the benchmark systems was in parts taken from [Wen+20] and slightly modified.

2.1 INTRODUCTION AND RELATED WORK

The main motivation for the work presented in this chapter lies in the widespread use of parametric models, e.g., in systems biology or neuroscience. [BKS14; PBP18] In this context, a Bayesian estimate of the system's parameter vector can give valuable insight about robustness or guide Bayesian model selection. However, obtaining an estimate of the parameter's posterior can be computationally costly. Most nonlinear ODEs do not have a closed form solution and thus, standard methods for statistical inference need to numerically integrate the dynamics every time the parameters are changed. This was the original motivation for the work of Calderhead, Girolami, and Lawrence [CGL09], who pioneer the use of Gaussian-process-based gradient matching. Their work builds on previous, spline-based approaches [Var82; Ram+07], introducing the use of Gaussian processes to obtain a fully probabilistic model suitable for Bayesian inference with final uncertainty estimates. To match the derivatives of dynamics model and GP, Calderhead, Girolami, and Lawrence [CGL09] propose to use a product of experts heuristic (PoE). This heuristic has become state of the art, being reused by the later extensions by Dondelinger et al. [Don+13] and Gorbach, Bauer, and Buhmann [GBB17]. However, it is also criticized in the work of Wang and Barber [WB14], who propose a different modeling paradigm. However, Macdonald, Higham, and Husmeier [MHH15] later demonstrated that the paradigm of Wang and Barber [WB14] suffers

from theoretical inconsistencies. While this establishes the PoE heuristics as the only remaining alternative, a similar theoretical analysis for the PoE approach has thus far been missing.

The need for such an analysis is further demonstrated by another peculiarity. Gorbach, Bauer, and Buhmann [GBB17] introduce a mean field variational inference approach to significantly reduce the run time of the Markov chain Monte Carlo schemes (MCMC) deployed by Calderhead, Girolami, and Lawrence [CGL09] and Dondelinger et al. [Don+13]. However, in addition to the expected significant decrease in run time, they also report a significant increase in accuracy.

The work presented in this chapter aims to tackle both of these challenges. First, the PoE heuristic is analyzed, which leads to the discovery of theoretical inconsistencies in most state of the art approaches. These theoretical inconsistencies are then mitigated by proposing a new theoretical framework, replacing the PoE heuristic completely. In a subsequent analysis, the accuracy improvements of Gorbach, Bauer, and Buhmann [GBB17] are found to be rooted in neither theory nor model, but in a careful choice of hyperparameters. This insight is combined with the new theoretical framework to develop a novel algorithm. The resulting algorithm improves state of the art algorithms in terms of accuracy and robustness, while providing a more computationally efficient sampling scheme that reduces its run time by roughly 35%.

2.2 PROBLEM SETTING AND NOTATION

In the following, we start by formalizing the problem. and giving an overview of related work, state-of-the-art methods and highlight on a high level the main contributions of our work in the context of Gaussian-process-based gradient matching.

As introduced in Section 1.1.1, we consider dynamical systems of the form

$$\dot{x}(t) = f(x(t), \boldsymbol{\theta}). \tag{2.1}$$

To keep the notation as simple as possible, we present all theory in the context of one-dimensional dynamical systems and thus drop the bold vector notation used in Equation (1.1) for the state vector $x(t)$ and its derivative. This does not restrict the applicability of our algorithm, since the contributions of multiple state dimensions could just be modelled independently, ultimately just summing up the corresponding terms in the

final objective. This is exactly the strategy deployed in the experiments in Section 2.7

Again, to keep the notation as simple as possible, we consider data coming from only one trajectory and thus, when describing our dataset, drop the dependence on the initial condition that we introduced in Section 1.1. This does not restrict the applicability of our algorithm, since the contributions of multiple trajectories could just be modelled independently.

Throughout this chapter, we will use some vector notation. First,

$$\boldsymbol{t} \coloneqq [t_1, \dots, t_{N_d}] \tag{2.2}$$

denotes the vector of all observation times. Similarly,

$$\boldsymbol{x} \coloneqq [x(t_1), \dots, x(t_{N_d})] \tag{2.3}$$

and

$$\dot{\boldsymbol{x}} \coloneqq [\dot{x}(t_1), \dots, \dot{x}(t_{N_d})] \tag{2.4}$$

denote the true state and its true derivative at those times. Finally, the observations are summarized in the vector

$$\boldsymbol{y} \coloneqq [y(t_1), \dots, y(t_{N_d})]. \tag{2.5}$$

These observations are assumed to have been created by distorting the true states $\boldsymbol{x}$ with i.i.d, additive Gaussian noise $\epsilon(t_i) \sim \mathcal{N}(0, \sigma^2)$, i.e.,

$$y(t_i) = x(t_i) + \epsilon(t_i), \qquad i = 1, \dots, N_d, \tag{2.6}$$

or equivalently

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \sigma) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{x}, \sigma^2 \boldsymbol{I}). \tag{2.7}$$

2.3 GP-BASED GRADIENT MATCHING - STATE OF THE ART

As mentioned in the previous chapter, gradient matching schemes aim to calculate two different values for $\dot{\boldsymbol{x}}$, one purely data-centered, and one involving the parametric system dynamics. While there is a lot of debate on how to match these derivatives and on what algorithms to use for inference, all state of the art, Gaussian-process-based gradient matching schemes share the same core, a Gaussian process employed as a smoother.

2.3.1 *Calculating Derivatives: Basic Modeling Choices*

The purpose of this Gaussian process smoother is to build a regression model mapping the time points to the corresponding state values. For this, one needs to choose an appropriate kernel function $k_{\boldsymbol{\phi}}(t_i, t_j)$, which is parametrized by the hyperparameters $\boldsymbol{\phi}$. Both, the choice of kernels as well as how to fit its hyperparameters is discussed in Section 2.5.

Once the kernel and its hyperparameters are fixed, the covariance matrix $\boldsymbol{C}_{\boldsymbol{\phi}}$, whose elements are given by $\boldsymbol{C}_{\boldsymbol{\phi}}(i,j) = k(t_i, t_j)$, can be constructed and used to define a standard zero mean Gaussian process prior on the states,

$$p(\boldsymbol{x} \mid \boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{x} \mid \mathbf{0}, \boldsymbol{C}_{\boldsymbol{\phi}}). \tag{2.8}$$

As differentiation is a linear operation, the derivative of a Gaussian process is again a Gaussian process. Using probabilistic calculus, this fact leads to a distribution over the derivatives conditioned on the states at the observation points:

$$p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\phi}) = \mathcal{N}(\dot{\boldsymbol{x}} \mid \boldsymbol{D}\boldsymbol{x}, \boldsymbol{A}). \tag{2.9}$$

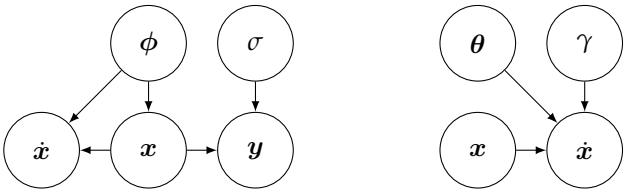
Additionally, a second model to obtain a distribution over the derivatives can be constructed by using the information provided by the differential equations. For known states and parameters, one can calculate the derivatives using equation (1.1). A potential modeling mismatch between the output of the ODEs and the derivatives of the GP model is accounted for by introducing isotropic Gaussian noise with standard deviation γ, leading to the following Gaussian distribution over the derivatives:

$$p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\theta}, \gamma) = \mathcal{N}(\dot{\boldsymbol{x}} \mid f(\boldsymbol{x}, \boldsymbol{\theta}), \gamma \boldsymbol{I}). \tag{2.10}$$

Thus, we have obtained two different generative models for the derivatives, summarized by Equations (2.9) and (2.10). The statistical dependencies of these models are further visualized in Figure 2.1.

2.3.2 *Merging the two Components Via the Product of Experts Heuristics*

The main goal of gradient matching is to use the observations $\boldsymbol{y}$, only present in the GP model, to infer the parameters $\boldsymbol{\theta}$, only present in the ODE model. However, combining the two models is not straight forward. In both models, the states $\boldsymbol{x}$ and the state's derivatives $\dot{\boldsymbol{x}}$ are present as random variables.



(a) Gaussian Process Model (b) ODE Model

FIGURE 2.1: Visualization of the modelled dependencies of the two base components of Gaussian-Process-based gradient matching.

For the states, a quick solution can be found. In the ODE model, x has only one child node and no parents. This is due to the fact that in the ODE model, we just condition on a not further specified distribution over x. To combine the two models, the most natural choice for the distribution over x in the ODE model is just to choose whatever distribution comes from the GP model. Thus, the two nodes representing x in the GP and the ODE models can be merged with no adaptations necessary.

Unfortunately, the derivatives are dependent variables in both models. Thus, handling them is not as straight forward, since we are now dealing with a random variable with two different distributions. Calderhead, Girolami, and Lawrence [CGL09] thus propose to deploy the product of experts heuristic. The main idea of the product of experts, originally introduced by Hinton [Hin02], is to infer the probability density of a variable by normalizing the product of multiple expert densities. This makes it directly applicable to this case, yielding

$$p(\dot{x} \mid x, \phi, \theta, \gamma) \propto p(\dot{x} \mid x, \phi) p(\dot{x} \mid x, \theta, \gamma). \tag{2.11}$$

The idea of this approach is that the resulting density only assigns high probability if both experts assign high probabilities. Hence, it considers only cases in which both experts agree. It is thus based on the intuition that the true θ should correspond to a model that agrees both with the ODE model and the observed data. While this is intuitively well-motivated, we will show that the product of experts heuristic leads to theoretical difficulties and offer an alternative in Section 2.4.

2.3.3 *State of the Art Inference*

Building on this generative model, various inference schemes have been developed and analyzed.

Inference with Markov Chain Monte Carlo (MCMC)

Calderhead, Girolami, and Lawrence [CGL09] combine the product of experts with equations (2.7), (2.8) and (2.9) and some suitable prior over $\boldsymbol{\theta}$ to obtain a joint distribution $p(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\theta}, \boldsymbol{\phi}, \sigma \mid \boldsymbol{y})$. After integrating out $\dot{\boldsymbol{x}}$, which can be done analytically, since Gaussian processes are closed under linear operators (and using some proportionality arguments), a sampling scheme was derived that consists of two MCMC steps. First, the hyperparameters of the GP, $\boldsymbol{\phi}$ and σ, are sampled from the conditional distribution $p(\boldsymbol{\phi}, \sigma | \boldsymbol{y})$. Then, a second MCMC scheme is deployed to infer the parameters of the ODE model, $\boldsymbol{\theta}$ and γ, by sampling from the conditional distribution $p(\boldsymbol{\theta}, \gamma | \boldsymbol{x}, \boldsymbol{\phi}, \sigma)$.

Dondelinger et al. [Don+13] then reformulated the approach by directly calculating the joint distribution

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}, \gamma, \sigma) \propto \\ p(\boldsymbol{\theta}) \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}) \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{x}, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{D}\boldsymbol{x}, \boldsymbol{A} + \gamma \boldsymbol{I}), \quad (2.12)$$

where the proportionality is meant to be taken w.r.t. the latent states $\boldsymbol{x}$ and the ODE parameters $\boldsymbol{\theta}$. Here $p(\boldsymbol{\theta})$ denotes some prior on the ODE parameters. This approach was named Adaptive Gradient Matching (AGM).

Inference Using a Mean Field Variational Approximation

The main idea of Variational Gradient Matching (VGM), introduced by Gorbach, Bauer, and Buhmann [GBB17], is to substitute the MCMC inference scheme of AGM with a mean field variational inference approach, approximating the density in Equation (2.12) with a fully factorized Gaussian over the states $\boldsymbol{x}$ and the parameters $\boldsymbol{\theta}$. To obtain analytical solutions, the functional form of the ODEs is restricted to locally linear functions that can be written as

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_i \theta_i \prod_{j \in \mathcal{M}_i} x_j \qquad \text{where} \quad \mathcal{M}_i \subseteq \{1, ..., K\}. \quad (2.13)$$

As perhaps expected, VGM is magnitudes faster than the previous sampling approaches. However, despite being a variational approach, VGM was

also able to provide significantly more accurate parameter estimates than both sampling-based approaches of Calderhead, Girolami, and Lawrence [CGL09] and Dondelinger et al. [Don+13]. In Section Section 2.7, we provide justification for these surprising performance differences.

2.3.4 *Gradient Matching Without Product of Experts*

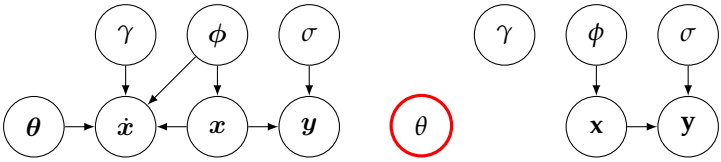
Up to our knowledge, before the work presented in this chapter, there was only one approach that tried to do Gaussian-Process-based gradient matching without the product of experts formulation, namely the work of Wang and Barber [WB14]. However, this approach had its own theoretical problems, which were intensively discussed in [MHH15].

2.4 THEORY

In this section, we will present the main theoretical contributions of this chapter. At the start of this section, we investigate the PoE heuristics. The challenges arising from this heuristics are highlighted with two arguments, one based on graphical models and one based on the original mathematical derivation by Calderhead, Girolami, and Lawrence [CGL09]. To fix these issues, we then provide a new graphical model and finally demonstrate how to do inference in this new model.

2.4.1 *Analysis Of The Product Of Experts Approach*

As previously stated, Calderhead, Girolami, and Lawrence [CGL09], Dondelinger et al. [Don+13] and Gorbach, Bauer, and Buhmann [GBB17] all use a product of experts to obtain $p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \phi, \boldsymbol{\theta}, \gamma)$ as stated in Equation (2.11).



(a) After Product of Experts (b) After Marginalizing Out $\dot{\mathbf{x}}$

FIGURE 2.2: The product of experts approach illustrated with graphical models. After marginalization of $\dot{\boldsymbol{x}}$, the parameters $\boldsymbol{\theta}$ we would like to infer are independent of the observations $\boldsymbol{y}$.

Figure 2.2 depicts what is happening if the product of experts approach is applied in the gradient matching framework. Figure 2.2a depicts the graphical model after the two models have been merged using the product of experts heuristic of Equation (2.11). Using the distribution over $\boldsymbol{x}$ of the Gaussian process model represented in Figure 2.1a as a prior for the $\boldsymbol{x}$ in the ODE response model represented in Figure 2.1b, effectively leads to merging the two nodes representing $\boldsymbol{x}$. Furthermore, the product of experts heuristic implies by its definition that after applying Equation (2.11), $\dot{\boldsymbol{x}}$ is only depending on $\boldsymbol{x}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$ and γ.

In the graphical model in Figure 2.2a, the problem is already visible. The ultimate goal of merging the two graphical models is to create a probabilistic link between the observations $\boldsymbol{y}$ and the ODE parameters $\boldsymbol{\theta}$. However, the newly created connection between these two variables is given via $\dot{\boldsymbol{x}}$, which has no outgoing edges and of which no observations are available. Marginalizing out $\dot{\boldsymbol{x}}$ as proposed in the traditional approaches consequently leads to the graphical model in Figure 2.2b. As there is no directed path connecting other variables via $\dot{\boldsymbol{x}}$, all the different components are now independent. Consequently, the posterior over $\boldsymbol{\theta}$ is now given by the prior we put on $\boldsymbol{\theta}$ in the first place.

This problem can further be illustrated by looking at the mathematical derivations in the original paper of Calderhead, Girolami, and Lawrence [CGL09]. After carefully calculating the omitted but necessary normalization constants, the last equation in the third chapter is equivalent to stating

$$p(\boldsymbol{\theta}, \gamma \mid \boldsymbol{x}, \boldsymbol{\phi}, \sigma) = \int p(\boldsymbol{\theta}) p(\gamma) p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\theta}, \gamma, \boldsymbol{\phi}, \sigma) d\dot{\boldsymbol{x}}. \tag{2.14}$$

It is clear that this equation should simplify to

$$p(\boldsymbol{\theta}, \gamma \mid \boldsymbol{x}, \boldsymbol{\phi}, \sigma) = p(\boldsymbol{\theta}) p(\gamma). \tag{2.15}$$

Thus, one could argue that any links that are not present in the graphical model of Figure 2.2b but found by Calderhead, Girolami, and Lawrence [CGL09] and reused in Dondelinger et al. [Don+13] and Gorbach, Bauer, and Buhmann [GBB17] were created by improper normalization of the density $p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\theta}, \gamma, \boldsymbol{\phi}, \sigma)$.

2.4.2 *Adapting The Original Graphical Model*

Despite these technical difficulties arising from the PoE heuristic, the approaches provide good empirical results and have been used in practice,

e.g., by Babtie, Kirk, and Stumpf [BKS14]. In what follows, we derive an alternative model and mathematical justification for Equation (2.12) to provide a theoretical framework explaining the good empirical performance of Gaussian-process-based gradient matching approaches, especially from Gorbach, Bauer, and Buhmann [GBB17], which uses only weak or nonexistent priors.

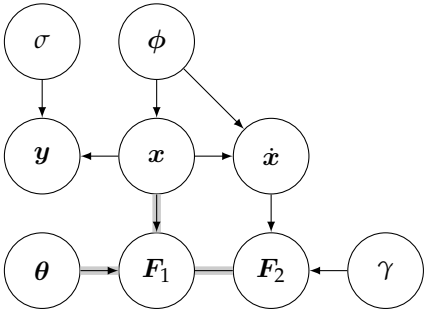


FIGURE 2.3: Alternative probabilistic model without PoE heuristic. Gray shaded connections are used to indicate a deterministic relationship.

The graphical model shown in Figure 2.3 offers an alternative approach to the product of experts heuristic. The top two layers are equivalent to a GP prior on the states, the induced GP on the derivatives and the observation model, as shown in Figure 2.1a.

The interesting part of the new graphical model is the bottom layer. Instead of adding a second graphical model like in Figure 2.1b to account for the ODE response, two additional random variables are introduced.

$\boldsymbol{F}_1$ is the deterministic output of the ODEs, assuming the values of $\boldsymbol{x}$ and $\boldsymbol{\theta}$ are given, i.e., $\boldsymbol{F}_1 = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$. The deterministic nature of this equation is represented as a Dirac delta function, as

$$p(\boldsymbol{F}_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) = \delta(\boldsymbol{F}_1 - \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{x})). \tag{2.16}$$

If the GP model were able to capture the true states and true derivatives perfectly, this new random variable should be equivalent to the derivatives of the GP, i.e., $\boldsymbol{F}_1 = \dot{\boldsymbol{x}}$. However, to compensate for a potential model mismatch and slight errors of both GP states and GP derivatives, this condition is relaxed to

$$\mathbf{F}_1 = \dot{\boldsymbol{x}} + \boldsymbol{\epsilon} =: \boldsymbol{F}_2, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \gamma \boldsymbol{I}). \tag{2.17}$$

In the graphical model, this intuitive argument is encoded via the random variable $\boldsymbol{F}_2$. Given the $\dot{\boldsymbol{x}}$ provided by the GP model, Gaussian noise with standard deviation γ is added to create $\boldsymbol{F}_2$, whose probability density can thus be described as

$$p(\boldsymbol{F}_2 \mid \dot{\boldsymbol{x}}, \gamma) = \mathcal{N}(\boldsymbol{F}_2 \mid \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}). \tag{2.18}$$

The equality constraint given by equation (2.17) is represented in the graphical model by the undirected edge between $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$. When doing inference, this undirected edge is incorporated in the joint density via a Dirac delta function $\delta(\boldsymbol{F}_2 - \boldsymbol{F}_1)$. Thus, the joint density of the graphical model represented in Figure 2.3 can be written as

$$\begin{aligned} &p(\boldsymbol{x},\dot{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{\theta} \mid \boldsymbol{\phi}, \sigma, \gamma) = \\ &\quad p(\boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{\phi}) p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\phi}) p(\boldsymbol{y}|\boldsymbol{x}, \sigma) p(\boldsymbol{F}_1 \mid \boldsymbol{\theta}, \boldsymbol{x}) p(\boldsymbol{F}_2 \mid \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}) \delta(\boldsymbol{F}_1 - \boldsymbol{F}_2). \end{aligned} \tag{2.19}$$

2.4.3 *Inference In The New Model*

Given all the definitions in the previous section, inference can now be directly performed without the need for additional heuristics. The result is a theoretically sound justification of the formula that forms the basis of the main results of Calderhead, Girolami, and Lawrence [CGL09], Dondelinger et al. [Don+13] and Gorbach, Bauer, and Buhmann [GBB17]:

Theorem 1 *Given the modeling assumptions summarized in the graphical model in Figure 2.3,*

$$\begin{aligned} &p(\boldsymbol{y}, \boldsymbol{x},\boldsymbol{\theta}, \boldsymbol{\phi}, \gamma, \sigma) \propto \\ &\qquad p(\boldsymbol{\theta}) \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}) \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{x}, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{D}\boldsymbol{x}, \boldsymbol{A} + \gamma \boldsymbol{I}). \end{aligned} \tag{2.20}$$

The proof can be found in the supplementary material, Appendix A.1.1.

2.5 HYPERPARAMETERS AND PREPROCESSING

Given the graphical model of Figure 2.3, we still need to determine how to set up the Gaussian process, i.e., how to choose an appropriate kernel $k_{\boldsymbol{\phi}}(t_i, t_j)$, including its hyperparameters $\boldsymbol{\phi}$, and values for the observation noise's standard deviation σ. In this section, we will give an overview of

how this problem has been handled in the context of GP-based gradient matching in the literature. Then, we will elucidate some practical aspects that are crucial to the performance of our algorithm.

2.5.1 *Hyperparameter and kernel selection*

For both the hyperparameters ϕ and the functional form of k there exist many possible choices. Even though the exact choice might not be too important for consistency guarantees in GP regression [CS07], this choice directly influences the number of observations that are needed for reasonable performance. While there exist some interesting approaches to learn the kernel directly from the data, e.g., the works of Duvenaud et al. [Duv+13] and Gorbach et al. [Gor+17], these methods can not be applied due to the very small number of observations of the systems considered in this paper. As in previous approaches, the kernel functional form is thus restricted to simple kernels with few hyperparameters, whose behaviors have already been investigated by the community, e.g., in the kernel cookbook by Duvenaud [Duv14]. Once a reasonable kernel is chosen, it is necessary to fit the hyperparameters. To fit hyperparameters, the choice of an appropriate scheme strongly depends on the amount of expert knowledge available.

No expert knowledge - Maximizing the data likelihood

For a Gaussian proces model it is possible to analytically calculate the marginal likelihood of the observations $\mathbf{y}$ given the evaluation times $\mathbf{t}$ and hyperparameters ϕ and σ [Ras04a], as

$$\log(p(\boldsymbol{y} \mid \boldsymbol{t}, \phi, \sigma)) = -\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{C}_\phi + \sigma\boldsymbol{I})^{-1} - \frac{1}{2}\log|\boldsymbol{C}_\phi + \sigma\boldsymbol{I}| - \frac{N_d}{2}\log 2\pi, \quad (2.21)$$

where σ is the GPs estimate for the standard deviation of the observation noise and N_d is the number of observations, as introduced at the beginning of this chapter.

To fit the GP model to the data, Equation (2.21) can be maximized w.r.t. ϕ and σ, without incorporating any prior knowledge. It is important to note that this method is completely independent of the ODEs one would like to analyze. The choice of hyperparameters will only depend on the observations of the states $\boldsymbol{y}$. Thus, it is the most agnostic way of fitting hyperparameters. This is also the method we will deploy in FGPGM, for reasons that will become clear later in this section.

Some expert knowledge available - Concurrent optimization

In AGM of Dondelinger et al. [Don+13], the hyperparameters are not calculated independently of the ODEs. Instead, a hyperprior is defined (i.e., a prior over the hyperparameters) and their posterior distribution is determined simultaneously with the posterior distribution over states and parameters by sampling from equation (2.12).

This approach has several drawbacks. As we shall see in section 2.7, its empirical performance is significantly depending on the hyperpriors. Furthermore, optimizing the joint distribution given equation (2.12) requires calculating the inverse of the covariance matrices $\boldsymbol{C}_{\phi}$ and $\boldsymbol{A}$, which has to be done again and again for each new set of hyperparameters. Due to the computational complexity of matrix inversion, this is significantly slowing down the inference of $\boldsymbol{\theta}$.

For these reasons, if strong prior knowledge about the hyperparameters is available, it might be better to incorporate it into maximizing the data likelihood. There, it could be easily be incorporated by including a hyperprior term to regularize the likelihood of Equation (2.21).

Reliable expert knowledge available - Manual tuning

In the variational inference approach of Gorbach, Bauer, and Buhmann [GBB17], the hyperparameters are assumed to be provided by an expert. If such expert knowledge is available, it should definitely be used since it can improve the accuracy drastically. As we shall see in our experiments in Section 2.7, this choice of hyperparameters is the main reason why the variational approach was able to outperform the MCMC-based approaches.

2.5.2 *Accounting for Normalization and Standardization*

To provide a fairer comparison between VGM of Gorbach, Bauer, and Buhmann [GBB17] and AGM of Dondelinger et al. [Don+13], the hyperparameter learning must involve an equal amount of expert knowledge. The most natural choice would be the maximization of the data likelihood, however, just maximizing the objective of Equation (2.21) using the prior defined in Equation (2.8) will lead to bad results.

Zero mean observations

The main reason for that is the fact that the zero mean assumption in equation (2.21) is a very strong regularization, especially for small data

sets. Furthermore, as its effect directly depends on the distance of the true values to zero, it will regularize differently for different state dimensions in multidimensional systems, further complicating the problem. To alleviate this problem, it is common to manipulate the observations such that they have zero mean. [Ras04b]

This procedure can be directly incorporated into the joint density given by equation (2.12). It should be noted that for multidimensional systems this joint density will factorize over each state k, whose contribution will be given by

$$p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}, \phi, \gamma, \sigma) \propto \mathcal{N}(\tilde{\boldsymbol{x}} | \boldsymbol{0}, \boldsymbol{C}_\phi) \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{x}, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{D}\tilde{\boldsymbol{x}}, \boldsymbol{A} + \gamma \boldsymbol{I}), \quad (2.22)$$

where

$$\tilde{\boldsymbol{x}} \coloneqq \boldsymbol{x} - \mu_y \boldsymbol{1}.$$

using μ_y to denote the mean of the observations and $\boldsymbol{1}$ to denote a vector of ones with appropriate length.

It should be noted that we cannot just substitute $\tilde{\boldsymbol{x}}$ for $\boldsymbol{x}$ in Equation (2.18), since the ODEs still need the original states $\boldsymbol{x}$ as inputs. While this makes notation a bit cumbersome, normalization is of crucial importance. In the simulations, this made a difference for all systems.

Standardized states

For multidimensional systems, normalization is not enough, since the states of different dimensions might be different in magnitude. If one were to use the same hyperparameters ϕ for all dimensions, then a deviation $(\boldsymbol{F} - \boldsymbol{D}\tilde{\boldsymbol{x}}) = 10^{-4}$ would contribute equally to the change in overall probability, independently of whether the states of this dimension $\tilde{\boldsymbol{x}}$ are of magnitude 10^{-8} or 10^3. Thus, small relative deviations from the mean of states with large values will lead to stronger changes in the overall probability than large relative deviations of states with small values. This is not a desirable property, which can be partially alleviated by calculating a new set of hyperparameters for each state. However, this problem can be completely nullified by standardizing the data $\boldsymbol{y}$. Thus, the contribution of one dimension is now given by

$$p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}, \phi, \gamma, \sigma) \propto \quad (2.23)$$
$$\mathcal{N}\left(\frac{1}{\sigma_y}\tilde{\boldsymbol{x}} \middle| \boldsymbol{0}, \boldsymbol{C}_\phi\right) \mathcal{N}\left(\frac{1}{\sigma_y}\boldsymbol{y} \middle| \frac{1}{\sigma_y}\boldsymbol{x}, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\frac{1}{\sigma_y}\boldsymbol{f}_k(\boldsymbol{x}, \boldsymbol{\theta}) \middle| \frac{1}{\sigma_y}\boldsymbol{D}\tilde{\boldsymbol{x}}, \boldsymbol{A} + \gamma \boldsymbol{I}\right),$$

where

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \mu_y \boldsymbol{1},$$

and σ_y is the standard deviation of the observations $\boldsymbol{y}$.

2.6 FGPGM - FAST GAUSSIAN PROCESS BASED GRADIENT MATCHING

After incorporating standardization and normalization into the preprocessing, maximizing the marginal log likelihood of the observations can be used to infer hyperparameters for VGM as well. Since this is a modification to the expert-informed hyperparameter tuning scheme of the original publication, we call the modified algorithm MVGM. This should provide a fairer comparison between the variational MVGM and the MCMC-based AGM.

However, MVGM is still outperforming AGM significantly in the experiments shown in Section 2.7. This suggests that the concurrent optimization of $\boldsymbol{\theta}$, $\boldsymbol{x}$ and $\boldsymbol{\phi}$ suggested by Dondelinger et al. [Don+13] is hurting the performance of the overall algorithm. Based on this insight, we propose the sequential approach shown in Algorithm 1. In a first step, the Gaussian process model is fit to the standardized data by calculating the hyperparameters via Equation (2.21). Then, the states $\boldsymbol{x}$ and ODE parameters $\boldsymbol{\theta}$ are inferred using a one chain MCMC scheme on the density given by equation (2.12). Not that in the algorithmic description, we explicitly used the variable k to denote the state dimension. Equations (2.12) and (2.21) should thus be understood as a sum over the independent contributions of each state dimension.

2.7 EXPERIMENTS

To demonstrate the empirical performance of the newly set up FGPGM, we investigate its performance by benchmarking it against AGM and MVGM in this section. For all experiments involving AGM, the R toolbox deGradInfer [MD17] published alongside [Mac17] was used. The toolbox contains code to run two experiments, namely Lotka Volterra and Protein Transduction. Both of these systems are used in this paper, as they are the two standard benchmark systems used in all previous publications. It should be noted however that the applicability of FGPGM is not restricted to these systems. Unlike AGM, FGPGM refrains from using hard to motivate hyperpriors and our publicly available implementation can easily be adapted to new settings.

All algorithms were provided with one hyperparameter. While the toolbox of AGM had to be provided with the true standard deviation of the observation noise, MVGM and AGM were provided with γ. The γ was determined by testing eight different values logarithmically spaced between 1 and 10^{-4} and comparing the results based on observation fit.

Algorithm 1 Fast Gaussian Process Based Gradient Matching (FGPGM)

Input: $\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \gamma, N_{MCMC}, N_{burnin}, \boldsymbol{t}, \sigma_s, \sigma_p$
Step 1: Fit GP model to data
for all $k \in K$ **do**
    $\boldsymbol{\mu}_{y,k} \leftarrow \text{mean}(\boldsymbol{y}_k)$
    $\sigma_{y,k} \leftarrow \text{std}(\boldsymbol{y}_k)$
    $\tilde{\boldsymbol{y}} \leftarrow (\boldsymbol{y_k} - \boldsymbol{\mu}_k)/\sigma_{y,k}$
    Find $\boldsymbol{\phi}_k$ and σ_k by maximizing $p(\tilde{\boldsymbol{y}} \mid \boldsymbol{t}, \boldsymbol{\phi}_k, \sigma_k)$ of Equation (2.21).
end for
Step 2: Infer $\boldsymbol{x}$ and $\boldsymbol{\theta}$ using MCMC
$\mathcal{S} \leftarrow \emptyset$
for $i = 1 \rightarrow N_{MCMC} + N_{burnin}$ **do**
    for each state and parameter **do**
        $\mathcal{T} \leftarrow \emptyset$
        Propose a new state or parameter value by adding a zero mean Gaussian increment with standard deviation σ_s or σ_p.
        Accept proposition based on the density of Equation (2.12).
        Add current value to $\mathcal{T}$.
    end for
    Add the mean of $\mathcal{T}$ to $\mathcal{S}$.
end for
Discard the first N_{burnin} samples of $\mathcal{S}$.
Return the mean of $\mathcal{S}$.
Return: $\mathbf{x}, \boldsymbol{\theta}$

To obtain a cleaner presentation, not all results are shown in the main body of the text. Additional plots can be found in the appendix, Section A.1.

2.7.1 *Benchmark Tasks*

In our experiments, we are using three systems, Lotka Volterra (LV), Protein Transduction (PT) and FitzHugh-Nagumo (FHN). To guarantee a fair comparison, we follow the established parameter settings as adopted among others by Calderhead, Girolami, and Lawrence [CGL09], Dondelinger et al. [Don+13], Gorbach, Bauer, and Buhmann [GBB17] and Wenk et al. [Wen+19], which we will introduce in the following.

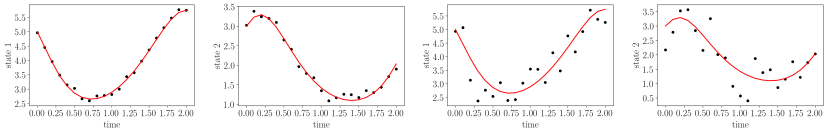
Lotka Volterra

The Lotka-Volterra predator-prey model was originally introduced by Lotka [Lot32] to describe population dynamics. It is a 2D system whose dynamics are determined by the ODEs

$$\dot{x}_1(t) = \quad \theta_1 x_1(t) - \theta_2 x_1(t) x_2(t) \tag{2.24}$$
$$\dot{x}_2(t) = -\theta_3 x_2(t) + \theta_4 x_1(t) x_2(t). \tag{2.25}$$

Using $\boldsymbol{\theta} = [2, 1, 4, 1]$ and initial conditions $\mathbf{x}(0) = [5, 3]$, the system is in a stable limit cycle with very smooth trajectories. The training data in this case consists of 20 equally spaced observations on the interval $[0, 2]$. It should be noted that the ODEs are linear in one state or parameter variable if all other state and parameter variables are kept constant. In the context of Gaussian process-based gradient matching, this means that the posterior marginals of the parameters and states are Gaussian distributed, which makes this system rather easy to solve. For the final data set, observations are created by adding i.i.d. Gaussian noise to the simulated ground truth. We investigate two cases, a low noise case with a noise standard deviation of 0.1 and a high noise case with a noise standard deviation of 0.5. For each case, an example is visualized in Figure 2.4.

Protein Transduction



(a) low noise - state 1 (b) low noise - state 2 (c) high noise - state 1 (d) high noise - state 2

FIGURE 2.4: Ground truth (red) and observations (black dots) of one example of a low and a high noise dataset of the Lotka Volterra dynamics. One data set contains 20 observations per state dimension of one trajectory.

The Protein Transduction model was originally introduced by Vyshemirsky

and Girolami [VG07] to model chemical reactions in a cell. It is a 5D system whose dynamics are described by the ODEs

$$\begin{aligned}
\dot{S} &= -\theta_1 S - \theta_2 SR + \theta_3 R_S \\
\dot{dS} &= \theta_1 S \\
\dot{R} &= -\theta_2 SR + \theta_3 R_S + \theta_5 \frac{R_{pp}}{\theta_6 + R_{pp}} \\
\dot{R}_S &= \theta_2 SR - \theta_3 R_S - \theta_4 R_S \\
\dot{R}_{pp} &= \theta_4 R_S - \theta_5 \frac{R_{pp}}{\theta_6 + R_{pp}}.
\end{aligned} \tag{2.26}$$

The parameters and initial conditions of this system were set respectively to $\boldsymbol{\theta} = [0.07, 0.6, 0.05, 0.3, 0.017, 0.3]$ and $\mathbf{x}(0) = [1, 0, 1, 0, 0]$. Due to the dynamics changing rapidly at the beginning of the trajectories, the training data is generated by sampling the system at $\mathbf{t} = [0, 1, 2, 4, 5, 7, 10, 15, 20, 30, 40, 50, 60, 80, 100]$. We investigate two cases, a low noise case with a noise standard deviation of 0.001 and a high noise case with a noise standard deviation of 0.01.

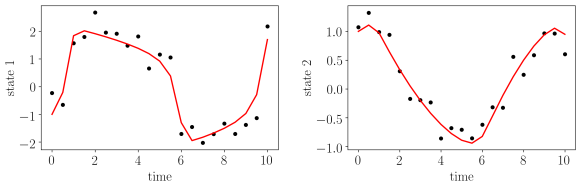
FitzHugh-Nagumo

The FitzHugh-Nagumo model was originally introduced by FitzHugh [Fit61] and Nagumo, Arimoto, and Yoshizawa [NAY62] to model the activation of giant squid neurons. It is a 2D system whose dynamics are described by the ODEs

$$\dot{V} = \theta_1 (V - \frac{V^3}{3} + R) \tag{2.27}$$

$$\dot{R} = \frac{1}{\theta_1}(V - \theta_2 + \theta_3 R). \tag{2.28}$$

Using $\boldsymbol{\theta} = [0.2, 0.2, 3]$ and initial conditions $\mathbf{x}(0) = [-1, 1]$, this system is in a stable limit cycle. However, the trajectories of this system are quite rough with rapidly changing lengthscales, which is a significant challenge for any smoothing-based scheme. Furthermore, both V and θ_1 appear nonlinearly in the ODEs, leading to non-Gaussian posteriors. The dataset for this case consists of 20 equally spaced observations on the interval $[0, 10]$.We investigate two cases, a low noise case with a signal-to-noise ratio (SNR) of 100 and a high noise case with a SNR of 10. In the low noise case, the observations are barely distinguishable from the ground truth, the high noise case is visualized in Figure 2.5



(a) SNR 10 - state 1 (b) SNR 10 - state 2

FIGURE 2.5: Ground truth (red) and observations (black dots) of one example of a low and a high noise dataset of the FitzHugh-Nagumo dynamics for a SNR of 10. One data set contains 20 observations per state dimension of one trajectory.

2.7.2 *Evaluation*

The main goal of system identification is to find a function that fits the underlying ground truth. In the context of parameter inference, this means that we should not necessarily directly compare inferred ODE parameters. Instead, we propose to use the trajectory RMSE. For each parameter set, this metric measures how well the associated trajectories fit the ground truth. This provides a more natural quality measure, since RMSE in the space of parameters are meaningless without taking parameter sensitivity into account.

Definition 1 (Trajectory RMSE) *Let $\hat{\boldsymbol{\theta}}$ be the parameters estimated by an inference algorithm. Let $\mathbf{t}$ be the vector collecting the observation times. Define $\tilde{x}(t)$ as the trajectory one obtains by integrating the ODEs using the estimated parameters, but the true initial value, i.e.*

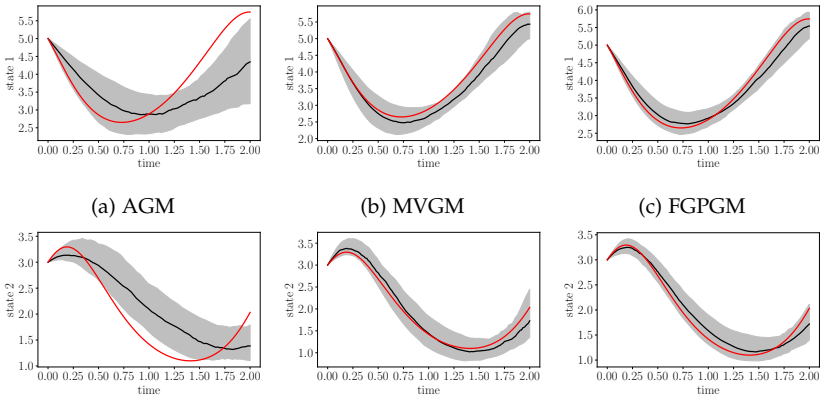
$$\tilde{x}(0) = \mathbf{x}^*(0) \tag{2.29}$$

$$\tilde{x}(t) = \int_0^t f(\tilde{x}(s), \hat{\boldsymbol{\theta}}) ds \tag{2.30}$$

and define $\tilde{\mathbf{x}}$ element-wise as its evaluation at observation times $\mathbf{t}$, i.e. $\tilde{x}_i = \tilde{x}(t_i)$. The trajectory RMSE is then defined as

$$tRMSE := \frac{1}{N} ||\tilde{\mathbf{x}} - \mathbf{x}||_2, \tag{2.31}$$

where $||.||_2$ denotes the standard Euclidean 2-norm.



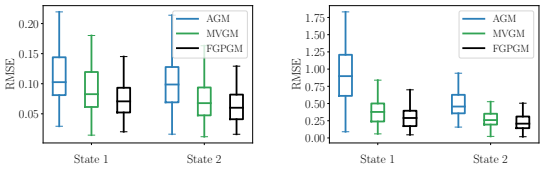
(a) AGM (b) MVGM (c) FGPGM

FIGURE 2.6: States after numerical integration of the inferred parameters in the high noise case of Lotka Volterra. Ground truth (red), median (black) and 75% quantiles (gray) over 100 independent noise realizations.

2.7.3 *Sampling vs Variational Inference - Lotka Volterra*

Lotka Volterra is by far the easiest benchmark considered here. However, it is the only standard benchmark satisfying Equation (2.13). Thus, it is needed to obtain a comparison against the variational MVGM. In Figure 2.6, we show the trajectories obtained by numerically integrating the inferred parameters for AGM, MVGM and FGPGM, with quantiles taken over 100 independent noise realizations. Following Dondelinger et al. [Don+13] and Gorbach, Bauer, and Buhmann [GBB17], a standard RBF kernel was used.

To obtain a more quantitive evaluation, we calculate the trajectory RMSE. It is shown in Figure 2.7, with the relative improvement of FGPGM shown in Table 2.1. In the next chapter, we will talk more about the comparison with AGM where we show more experiments. A comparison between FGPGM and AGM shows clearly a higher accuracy for the sampling-based FGPGM. This seems to indicate that - once the hyperparameters are taken out of the picture as tuning parameters - an MCMC-based approach might still outperform variational inference in terms of accuracy. This is important because, as we shall see in the next section, the functional form assumption of Equation (2.13) is severely restricting the applicability of MVGM, as we will see for the next benchmarks. Thus, FGPGM should be preferred if



(a) low noise (b) high noise

FIGURE 2.7: Trajectory RMSE obtained by numerical integration and comparison to the ground truth for Lotka Volterra. Boxplot with median (line), 50% (box) and 75% (whisker) quantiles over 100 independent noise realizations.

accuracy or flexibility in terms of the functional form of the dynamics is important.

	AGM	MVGM
LV low	35%	13%
LV high	62%	31%

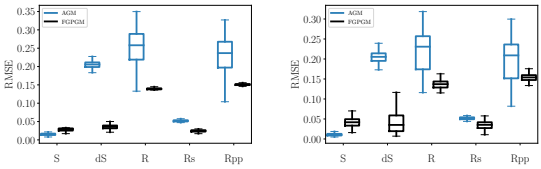
TABLE 2.1: Median reduction of state RMSE of FGPGM compared to AGM and MVGM as baseline on the Lotka Volterra system.

2.7.4 *Beyond Locally Linear Dynamics - Protein Transduction*

The protein transduction dynamics are a considerably harder benchmark. Previous work claimed it to be only weakly identifiable [Don+13], a claim we will come back to in Chapter 3. Due to its highly nonlinear terms, MVGM can no longer be applied.

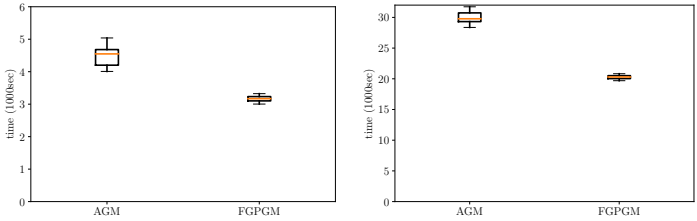
Thus, Figure 2.8 only shows the trajectory RMSE for AGM and FGPGM. Table 2.2 again quantifies the relative improvements of FGPGM over AGM. Finally, we compare the run time for both LV and PT in Figure 2.9. For all experiments, in line with the original work by Dondelinger et al. [Don+13], a sigmoid kernel was used.

FGPGM outperforms AGM both in terms of run time and in terms of accuracy. This seems to suggest that the sequential hyperparameter fitting scheme is not only improving computational efficiency, but also makes the algorithm more robust, if no strong hyperparameters are provided.



(a) low noise (b) high noise

FIGURE 2.8: Trajectory RMSE obtained by numerical integration and comparison to the ground truth for Protein Transduction. Boxplot with median (line), 50% (box) and 75% (whisker) quantiles over 100 independent noise realizations.



(a) run time LV (b) run time PT

FIGURE 2.9: FGPGM shows a clearly reduced run time compared to AGM, saving roughly 35% of computation time.

	AGM
PT low	50%
PT high	43%

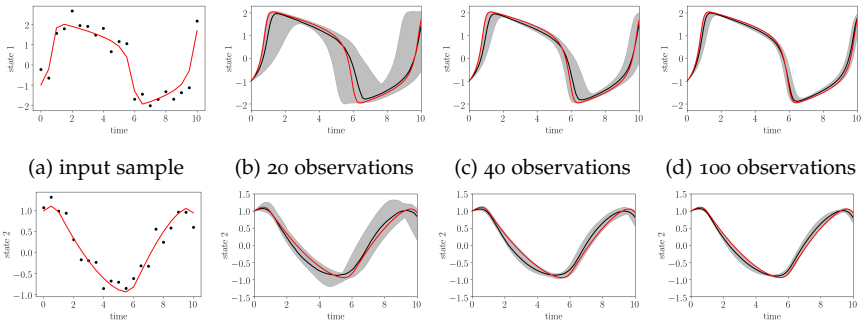
TABLE 2.2: Median reduction of state RMSE of FGPGM compared to AGM on the protein transduction system.

2.7.5 *Investigating Smoothness Bias - FitzHugh-Nagumo*

As can be seen in Figure 2.6, all GP-based gradient matching algorithms converge to parameter settings where the trajectories are smoother than the ground truth. While learning the hyperparameters in a pre-processing step clearly reduces this effect for FGPGM and MVGM, there is still some bias. If only few observations are available, the GP prior on the states tends to smooth out part of the system dynamics. This "smoothing bias" is then passed on to the ODE parameters in the gradient matching scheme.

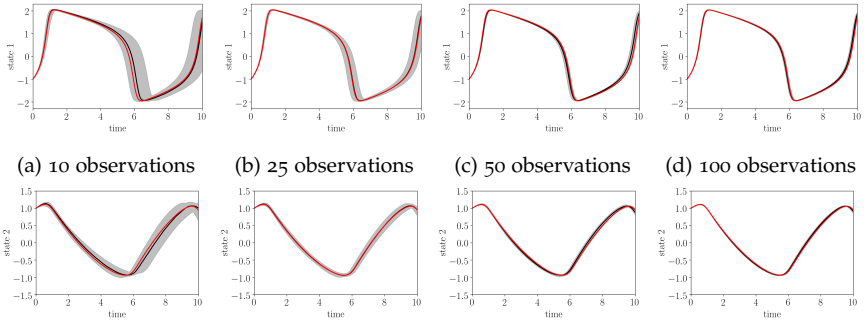
To investigate the practical importance of this effect, we evaluate FGPGM on the FHN system. Due to its highly nonlinear terms, the FHN system has notoriously fast changing dynamics, as can be seen in Figure 2.5. To account for the spikier behavior of the system, we used a Matérn52 kernel. γ was set to $3 \cdot 10^{-4}$.

As shown in Figure 2.10, the bias towards smoother trajectories is clearly visible. However, FGPGM finds parameters that tightly hug the ground truth. As to be expected, the smoothing bias gets smaller if more observations are added. Furthermore, increasing the SNR to a factor of 100 as shown in Figure 2.11 leads to excellent accuracy, even if we reduce the amount of observations to just 10. This is impressive, especially as FGPGM is a hyper-prior free, statistical method.



(a) input sample (b) 20 observations (c) 40 observations (d) 100 observations

FIGURE 2.10: One input sample and median plots of the numerically integrated states after parameter inference for the FHN system with SNR 10. Ground truth (red), median (black) and 75% quantiles (gray) over 100 independent noise realizations. In the most left plots, the black dots represent the noisy observations.



(a) 10 observations (b) 25 observations (c) 50 observations (d) 100 observations

FIGURE 2.11: Median plots of the numerically integrated states after parameter inference for the FHN system with SNR 10. Ground truth (red), median (black) and 75% quantiles (gray) over 100 independent noise realizations.

3

THE FREQUENTIST: ODE-INFORMED REGRESSION

While FGPGM works better than the comparisons in the context of Bayesian parameter inference, its performance still leaves some room for improvement. It can be seen, e.g., in Figure 2.10, the parameters inferred by FGPGM still exhibit some smoothing bias. Furthermore, in the theoretical derivations of Section 2.4, the generative model still requires an ad-hoc observation model with observation noise γ, which has to be tuned specifically for each model.

In this chapter, we will revisit the problem of parameter inference from a frequentist view point. Instead of aiming for posterior parameter distributions directly, we first aim for point estimates that are as accurate as possible. This allows for a more concise formulation of the gradient matching framework, where the ODEs are treated as a constraint instead of an additional generative model. As a result, we develop an algorithm that significantly improves empirical performance while getting rid of the last hyperparameter that cannot be directly inferred from data, i.e. γ. While we do not directly calculate uncertainties in this chapter, it should be noted that this could in principle be done in a frequentist setting as well. Here, one could deploy e.g. bootstrapping, which we suspect to be an efficient tool especially in the context of larger data sets.

The work in this chapter is heavily based on a conference publication [Wen+20] and an arxiv preprint [Ang+20], both of which were collaborations with other researchers. Some text and most of the figures were reused with permission to create this chapter.

3.1 PROBLEM SETTING AND RELATED WORK

In this chapter, we will exclusively consider the ODE-problem

$$\dot{x}(t) = f(x(t), \boldsymbol{\theta}), \tag{3.1}$$

that was introduced in Section 1.1.1 and briefly discussed in Section 1.2.1. As in the previous chapter, we will use a one-dimensional system to keep the notation concise.

Introducing constraints via the derivatives for a regression model is not a new idea. Lorenzi and Filippone [LF18] use this idea to regularize deep

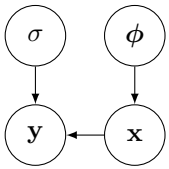


FIGURE 3.1: Generative model for standard Gaussian process regression. Given kernel hyperparameters ϕ and observation noise standard deviation σ, the probability densities for the states $\boldsymbol{x}$ and their noisy observations $\boldsymbol{y}$ are fully determined.

GP models, via a probabilistic constraint, that can also enforce inequality constraints for probabilistic inference. In the frequentist context, González, Vujačić, and Wit [GVW14] introduce this idea in the context of smoothers based on reproducing kernel hilbert spaces (RKHs). Their approach was later extended by Niu et al. [Niu+16] and applied to the ODE inference problem. Their approaches are naturally faster than any other discussed thus far that rely on MCMC and Gaussian processes. However, similar to FGPGM, they rely on several trade-off parameters that need to be tuned via cross-validation. In our experiments, we found that this is a key disadvantage in data-scarce environments.

Thus, our work is unique in the sense that we estimate both states and parameters while learning all all hyperparameters directly from data. While such hyperparameters are a nuisance for other algorithms that require tuning via cross-validation, the hyperparameters of ODIN can be used to indicate model-mismatch, as we shall demonstrate in our experiments.

3.2 METHODS

3.2.1 *Notation*

Throughout this chapter, we will use the notation introduced in Sections 2.2 and 2.3. In particular, we work with the observation noise standard deviation σ, the state and derivative vectors $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, the observation vector $\boldsymbol{y}$, the time vector $\boldsymbol{t}$, and the GP hyperparameters ϕ.

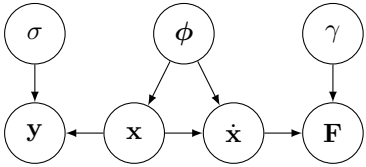


FIGURE 3.2: Generative model for GP regression with derivative observations $\boldsymbol{F}$, for which we use a Gaussian observation model with variance γ. Due to the GP prior, $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ are jointly Gaussian with known probability densities once the kernel hyperparameters $\boldsymbol{\phi}$ are determined.

3.2.2 *Generative Model*

In standard GP regression, we choose a kernel with some hyperparameters $\boldsymbol{\phi}$ and then use the generative model shown in Figure 3.1. Similar to the derivations of the generative model of FGPGM, this model is then extended by using the conditional $p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\phi})$ to include the state vector $\dot{\boldsymbol{x}}$. Unlike FGPGM however, we do not want to create a second generative model, leading to an overdetermined model. Instead, we start by assuming that we get access to some derivative observations

$$\boldsymbol{F} = \left[F(t_1, \ldots, F(t_{N_d})\right] \tag{3.2}$$

at the same time locations as the state observations $\boldsymbol{y}$. Similar to $\boldsymbol{y}$, $\boldsymbol{F}$ is a noisy version of the true derivatives of the dynamical system, distorted by additive, i.i.d. Gaussian noise

$$F(t_i) = \dot{\boldsymbol{x}}(t_i) + \delta(t_i), \qquad i = 1, \ldots, N_d, \tag{3.3}$$

where

$$\delta(t_i) \sim \mathcal{N}(0, \gamma). \tag{3.4}$$

Not that the γ here is not the same γ we introduced in the previous chapter. For FGPGM, γ was a slack parameter, that was necessary due to the heuristic used to match the two generative models. Here, γ is used as an observation noise standard deviation for the noise on the derivative observations $\boldsymbol{F}$. As of now, it is not tied to any ODE and just serves the purpose of quantifying the noise present in the (as of now) fictitious derivative observations $\boldsymbol{F}$.

Given these modeling assumptions, we obtain the generative model shown in Figure 3.2. To make notation more concise, we summarize the above statements in the two conditional probability densities

$$p(\dot{\boldsymbol{x}} \mid \boldsymbol{x}, \boldsymbol{\phi}) = \mathcal{N}(\dot{\boldsymbol{x}} \mid \boldsymbol{D}\boldsymbol{x}, \boldsymbol{A}), \tag{3.5}$$

with the matrices $\boldsymbol{D}$ and $\boldsymbol{A}$ being introduced in in Section 2.3, and

$$p(\boldsymbol{F} \mid \dot{\boldsymbol{x}}, \gamma) = \mathcal{N}(\boldsymbol{F} \mid \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}). \tag{3.6}$$

3.2.3 *ODIN - ODE-Informed Regression*

To avoid the problems associated with the two probabilistic models of traditional GP-based gradient matching, ODIN does not include the ODEs via a separate generative model. Instead, they are introduced at inference time in the form of *constraints*, essentially solving a constrained MAP problem.

We start with the joint density of the Gaussian process described in Figure 3.2, denoted by $p(\boldsymbol{y}, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{F} \mid \sigma, \gamma, \boldsymbol{\phi})$. As a result of the Gaussian observation model for $\boldsymbol{F}$, $\dot{\boldsymbol{x}}$ can be marginalized out analytically, leading to

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{F} \mid \sigma, \gamma, \boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{x}, \sigma^2 \boldsymbol{I})\, \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}})\, \mathcal{N}(\boldsymbol{F} \mid \boldsymbol{D}\boldsymbol{x}, \boldsymbol{A} + \gamma \boldsymbol{I}) \tag{3.7}$$

Assuming fixed values for σ, $\boldsymbol{\phi}$ and γ, this equation can be simplified by taking the logarithm, discarding all terms that do not explicitly depend on the states $\boldsymbol{x}$ and the derivative observations $\boldsymbol{F}$ and ignoring multiplicative factors to obtain

$$\tilde{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{F}, \boldsymbol{y}) = ||\boldsymbol{x}||^2_{\boldsymbol{C}_{\boldsymbol{\phi}}^{-1}} + ||\boldsymbol{x} - \boldsymbol{y}||^2_{\sigma^{-2}\boldsymbol{I}} + ||\boldsymbol{F} - \boldsymbol{D}\boldsymbol{x}||^2_{(\boldsymbol{A}+\gamma\boldsymbol{I})^{-1}}, \tag{3.8}$$

where $||\boldsymbol{u}||^2_{\boldsymbol{M}} \coloneqq \boldsymbol{u}^T \boldsymbol{M} \boldsymbol{u}$ is the norm of the vector $\boldsymbol{u}$ weighted by a positive-definite matrix $\boldsymbol{M}$.

The key mechanism behind ODIN lies in how we obtain values for $\boldsymbol{F}$. In principle, $\boldsymbol{F}$ could be marginalized out to recover standard GP regression. However, this is not desirable, as we would ignore the ODE information. Instead, ODIN includes the ODEs as additional constraints in the optimization problem: rather than keeping $\boldsymbol{F}$ completely flexible, we assume the existence of a parameter vector $\boldsymbol{\theta}$ that links the derivative observations to the ODEs. More formally,

$$\boldsymbol{x}, \boldsymbol{F} = \arg\min_{\boldsymbol{x}, \boldsymbol{\theta}} \tilde{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{F}, \boldsymbol{y}) \tag{3.9}$$

$$\text{s. t.} \quad \exists \boldsymbol{\theta} \quad \text{with} \quad \boldsymbol{f}(\boldsymbol{x}(t_i), \boldsymbol{\theta}) = \boldsymbol{F}_i \quad \text{for all } i. \tag{3.10}$$

As it turns out, these constraints can be incorporated in the optimization problem by directly substituting $\boldsymbol{F}$ with the corresponding contribution from the ODEs, leading to

$$\boldsymbol{x}, \boldsymbol{\theta} = \arg\min_{\boldsymbol{x}, \boldsymbol{\theta}} \mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}), \tag{3.11}$$

where $\mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}) := \tilde{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{y})$.

This constitutes the key idea behind ODIN. Instead of providing direct observations of the derivatives, we generate them via the ODEs after fixing the ODE parameters $\boldsymbol{\theta}$.

Similarly to classical frequentist methods [Var82; Niu+16], $\mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})$ penalizes divergence between the states $\boldsymbol{x}$ and the observations $\boldsymbol{y}$, as well as between the output of the ODEs, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$, and the derivatives estimated by the regressing GP; the regularization term avoids overfitting. Indeed, the representer theorem for derivatives [SHS01] shows close ties between the two approaches.

However, in sharp contrast to frequentist approaches, all trade-off parameters are naturally provided by the GP framework, once the hyperparameters ϕ and noise levels σ and γ are fixed. As we will see later, the absence of cross-validation for hyperparameter learning crucially improves accuracy in sparsely observed systems.

Algorithm 2 ODIN $-ODE-InformedRegression$

Input: $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(K)}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$
Step 1: State-independent GP regression
for all $k \in K$ **do**
    Standardize time $\boldsymbol{t}$ and observations $\boldsymbol{y}_k$.
    Fit $\boldsymbol{\phi_k}$ and σ_k using empirical Bayes, i.e. maximize $p(\boldsymbol{y}^{(k)} | \boldsymbol{t}, \boldsymbol{\phi}_k, \sigma_k)$.
    Initialize $\boldsymbol{x}_k$ using the mean $\boldsymbol{\mu}_k$ of the trained GP.
end for
Step 2: ODE Information Incorporation
Initialize $\boldsymbol{\theta}$ randomly.
Initialize $\gamma_1, \ldots, \gamma_K = 1.0$
Apply L-BFGS-B to solve the optimization problem (3.12) and obtain $\hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}}, \hat{\gamma}_1, \ldots, \hat{\gamma}_K$.
Return: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}}, \hat{\gamma}_1, \ldots, \hat{\gamma}_K$

3.2.4 *Derivative Observation Model*

Let us recall that we conceptually substitute the ODE outputs with derivative observations that are subjected to Gaussian noise, with variance γ. This allows for an intuitive but meaningful interpretation: during and after training, the ODE outputs and the GP derivative estimates can deviate from the ground truth and thus differ from each other. This divergence is accounted for by γ. In classical GP-based approaches [Wen+19; Don+13; GBB17], γ is treated as a random variable whose values are independent of the inference procedure; sometimes it is fixed a priori [Wen+19; GBB17]. However, we can expect that the divergence between ODEs and GP derivatives would be larger in the early steps of training, while it should decrease when the ODEs describe well to the ground truth. Thus, it is sensible to automatically adapt γ to reflect the current quality of the estimates.

The ODIN framework can be adjusted to reflect this reasoning. To obtain Equation (3.8), we implicitly assumed γ to be constant. If we rather treat it as an optimization parameter, the objective of Equation (3.11) changes to

$$\boldsymbol{x}, \boldsymbol{\theta}, \gamma = \arg\min_{\boldsymbol{x},\boldsymbol{\theta},\gamma} \mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}, \gamma) \tag{3.12}$$

where

$$\mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}, \gamma) = \mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}) + \log(\det(\boldsymbol{A} + \gamma \boldsymbol{I})). \tag{3.13}$$

If γ is part of the optimization procedure, the contribution of the normalization constant in Equation (3.7) can not be ignored when deriving the risk appearing in Equation (3.13). In practice, similarly to the log-determinant in standard GP regression, this term acts as an Occam's razor by preventing an excessive growth of γ if the GP derivatives and the ODE outputs differ significantly.

The final ODIN routine is summarized as Algorithm 2.

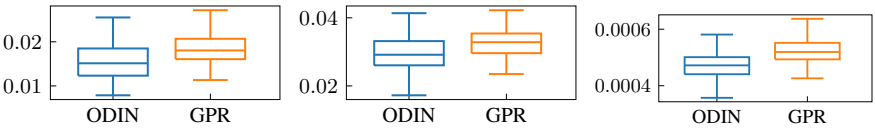
3.2.5 *Remarks*

Throughout this work, we assume to have access to observations $\boldsymbol{y}$ that are subjected to a Gaussian noise model with standard deviation σ. However, the Gaussian noise assumption is only needed when deriving the term $||\boldsymbol{x} - \boldsymbol{y}||^2_{\sigma^{-2}\boldsymbol{I}}$ in $\mathcal{R}$. Thus, it could be straightforward to accommodate for alternative noise models by adjusting the corresponding term in the risk formula. On the other side, things are less trivial for the Gaussian noise model (with variance γ) for the derivative observations. In our work, we

use this assumption to marginalize $\dot{x}$ analytically, a step that might not be as straight forward for different noise models.

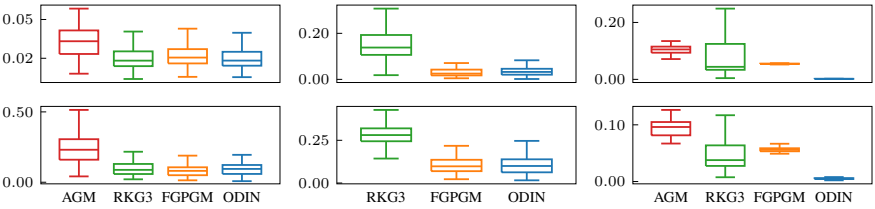
As demonstrates in the experiments section, in the case of perfect ODEs (i.e. when we know the true parametric form a priori), γ can in principle be set to zero; nevertheless, when that is not the case it provides an effective mechanism for detecting model mismatch and helps with the challenging problem of model selection.

3.3 EXPERIMENTS



(a) Lotka-Volterra (b) FitzHugh-Nagumo (c) Protein Transduction

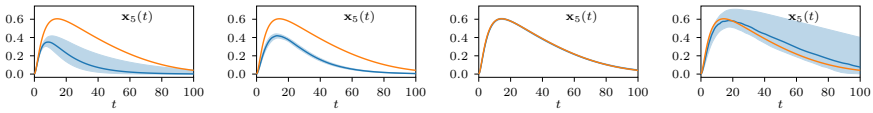
FIGURE 3.3: RMSE of state estimates using vanilla GP regression and ODIN on the benchmark systems (low noise case). While GPR can only access the noisy observations $\mathbf{y}$, ODIN considers the parametric form of $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$ (with no information about $\boldsymbol{\theta}$). This additional regularization contributes towards more accurate estimations.



(a) Lotka-Volterra (b) FitzHugh-Nagumo (c) Protein Transduction

FIGURE 3.4: Trajectory RMSE for parameter inference on the benchmark systems. The top row shows the low noise case with $\sigma = 0.1$ for LV, $SNR = 100$ for FHN and $\sigma = 0.001$ for PT. The bottom row shows the high noise case with $\sigma = 0.5$ for LV, $SNR = 10$ for FHN and $\sigma = 0.01$ for PT.

In this section, we demonstrate the versatility of ODIN and compare its performance to various state-of-the-art methods. We start by comparing its parameter inference capabilities on the three commonly used benchmark systems introduced in Section 2.7.1: the Lotka-Volterra (LV) predator-prey



(a) AGM (b) FGPGM (c) ODIN (d) RKG3

FIGURE 3.5: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the Protein Transduction system for $\sigma = 0.01$. The solid blue line is the median trajectory, while for clarity we shaded the area between the 25% and 75% quantiles. The orange trajectory represents the ground truth.

model; the FitzHugh-Nagumo (FHN) neuronal model; the chemical protein transduction (PT) system. In addition to state and parameter inference, we show how ODIN can be used for model selection, a missing feature for every comparison method here considered. Finally, we prove linear scaling behavior of ODIN in the state dimension K by investigating its performance on a high-dimensional, fourth benchmark system with up to 1000 states.

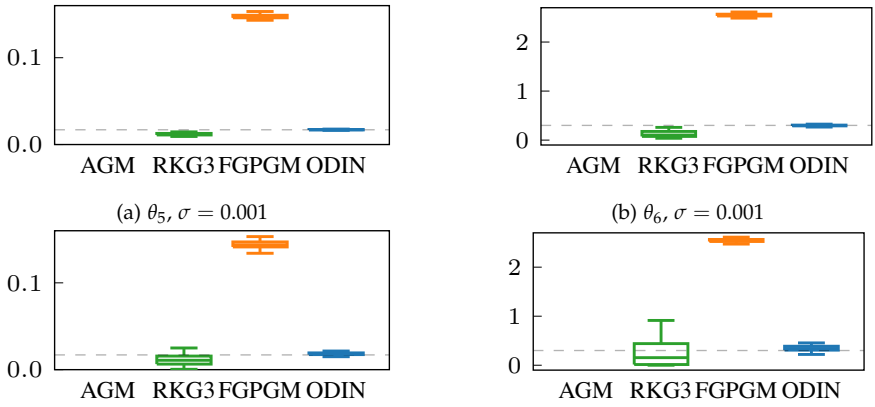
Wherever applicable, we will use the trajectory RMSE as comparison metric, as introduced in Definition (1).

Similar to the evaluation in the previous chapter, we evaluate the robustness of each algorithm w.r.t. different observation noise realizations. We always run 100 repetitions for every experimental setting. In each repetition, we keep the ground truth for states $\boldsymbol{x}^*$ and parameters $\boldsymbol{\theta}^*$ fixed and only sample the noise on $\boldsymbol{y}$. Results are then reported as quantiles over these 100 runs.

To complement the Bayesian parameter inference benchmarks of Chapter 2, we additionally compare against the frequentist RKG3. As previously mentioned in this chapter, its objectives are very closely related to ODIN via the generalized representer theorem [SHS01], but it requires additional cross-validation steps for trade-off parameters that come naturally in the ODIN framework.

3.3.1 *State and Parameter Inference*

In the parameter inference setting, the true parametric form of the dynamical system is assumed to be provided by a practitioner, derived through first principles or expert knowledge. Thus, together with the noisy observations $\boldsymbol{y}$, we have access to the true parametric form $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{\theta})$. The goal is



(a) θ_5, $\sigma = 0.001$

(b) θ_6, $\sigma = 0.001$

(c) θ_5, $\sigma = 0.01$

(d) θ_6, $\sigma = 0.01$

FIGURE 3.6: Parameter estimates for Protein Transduction for $\sigma = 0.001$ (a-b) and $\sigma = 0.01$ (c-d). Showing median, 50% and 75% quantiles over 100 independent noise realizations. The dashed line indicates the ground truth.

to recover the true states $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}^*$ at observation time. While smoothing is important, estimating $\boldsymbol{\theta}^*$ is of greater practical importance.

For all comparisons, implementations provided by the respective authors are used. In accordance to the gradient matching literature, evaluations include both a low and a high noise setting for every system.

As shown by Solak et al. [Sol+03], including direct observations of $\boldsymbol{F}$ can improve the accuracy of GP regression. ODIN does not have access to such observations, but it leverages the parametric form of the ODEs as a regularizer when performing state inference. As can be seen in Figure 3.3, this regularization actually improves the estimates of the states. This fact motivates a key difference to [CGL09], who propose to first fit the states using GPR and then perform gradient matching while keeping the states fixed. In the following, we show how ODIN can learn reliable parameters, improving the current state of the art in terms of accuracy and run time.

3.3.1.1 *Accuracy*

In Figure 3.4 we compare the trajectory RMSE for the three benchmark systems. While the total tRMSE is an effective indicator for the overall performance, we also include the state-wise tRMSE in the appendix. Unfor-

	AGM[s]	RKG3[s]	FGPGM[s]	ODIN[s]
LV, $\sigma = 0.1$	4548.0 ± 453.8	79.0 ± 19.0	3169.5 ± 90.1	$\mathbf{13.4 \pm 5.1}$
LV, $\sigma = 0.5$	4545.0 ± 558.5	76.5 ± 15.8	3187.5 ± 340.9	$\mathbf{11.4 \pm 5.1}$
FHN, $SNR = 100$	/	74.5 ± 14.3	8678.0 ± 482.7	$\mathbf{5.8 \pm 3.5}$
FHN, $SNR = 10$	/	77.5 ± 12.3	8677.0 ± 487.8	$\mathbf{4.4 \pm 3.8}$
PT, $\sigma = 0.001$	29776.5 ± 4804.7	469.0 ± 21.6	20291.5 ± 435.3	$\mathbf{8.9 \pm 1.5}$
PT, $\sigma = 0.01$	30493.0 ± 1470.4	480.0 ± 42.0	20437.0 ± 713.2	$\mathbf{20.6 \pm 3.75}$

TABLE 3.1: Median and standard deviation of computation time (in seconds) for parameter inference over 100 independent noise realizations.

tunately, AGM was unstable on FitzHugh-Nagumo despite serious hyperprior tuning efforts on our side. We thus do not have any results for this case. To help visualizing the raw numbers obtained by the tRMSE, we also report in Figure 3.5 the trajectories obtained by numerically integrating the inferred parameters. While here we report only one state for the high noise case of Protein Transduction, a full set of plots can be found in the appendix.

3.3.1.2 *Run time*

In Table 3.1, we list the median training times (in seconds) and the corresponding standard deviation of all algorithms on the three parameter inference benchmark systems. It is evident (and not unexpected) that the optimization-based algorithms ODIN and RKG3 are orders of magnitude faster than the MCMC-based FGPGM and AGM. Furthermore, the need for cross-validation schemes in RKG3 seems to increase its run time roughly by an order of magnitude when compared to ODIN.

3.3.1.3 *Identifiability*

While both LV as well as FHN models are relatively simple, Protein Transduction (PT) still represents a considerable challenge. Amongst others, both Dondelinger et al. [Don+13] and Wenk et al. [Wen+19] claim that the two parameters θ_5 and θ_6 are only weakly identifiable. However, a quick experiment with different numerical values for those ODE parameters shows that they are actually identifiable. Indeed, neither RKG3 and ODIN seem to suffer from identifiability problems. For ODIN, this can be attributed to two key differences, the inference scheme and the flexible γ. Both AGM and FGPGM ultimately return the posterior mean of the parameter marginals.

In Figure 3.7, we show example marginals for fixed γ. While these distributions are Gaussian-shaped for Lotka-Volterra, they are much wilder for PT. If we were to keep the γ fixed, ODIN would converge to an optimum instead of an expectation, which might be more appropriate in a multi-modal setting. However, ODIN does not keep γ fixed. Instead, its γ evolves during optimization according to the quality of the current parameters estimation, leading to an overall smoother inference. Consequently, the final parameter estimates are significantly more accurate (see Figure 3.6). For AGM, while the ratio between θ_5 and θ_6 is fairly stable and reasonably not far from the correct number, the absolute parameter values have median magnitudes of roughly 10^{12}: thus they do not appear in this figure.

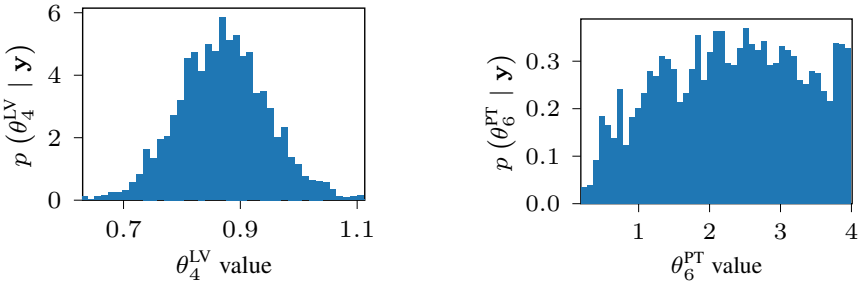


FIGURE 3.7: Parameter marginal distributions of θ_4 of Lotka-Volterra and θ_6 of Protein Transduction for one sample rollout with fixed γ. While the LV marginal is nicely Gaussian, the PT marginal is much wilder.

	$\mathcal{M}_{1,1}$	$\mathcal{M}_{0,1}$	$\mathcal{M}_{1,0}$	$\mathcal{M}_{0,0}$
γ_1	$10^{-6} \pm 0.00$	3.01 ± 0.23	$10^{-6} \pm 0.00$	3.03 ± 0.24
γ_2	$10^{-6} \pm 0.04$	$10^{-6} \pm 0.00$	1.51 ± 0.31	1.53 ± 0.35

TABLE 3.2: Median and standard deviation of γ for different model misspecifications of the Lotka-Volterra system and 100 independent noise realizations.

3.3.1.4 *Robustness*

Besides accurate parameter estimates, ODIN also exhibits more contained variance, especially compared to AGM and RKG3. This is a direct consequence of the underlying GP structure, which enables efficient and stable

calculation of all parameters. Furthermore, a flexible γ seems to smooth out the optimization surface, avoiding the rugged landscapes reported by Dondelinger et al. [Don+13].

3.3.1.5 *Priors*

While in a Bayesian inference setting it is common to introduce a prior over $\boldsymbol{\theta}$, our graphical model in Figure 3.2 does not treat $\boldsymbol{\theta}$ as a random variable. In a practical setting, we might not even know the parametric form of the ODEs: it thus seems quite difficult to justify the use of a prior. However, it should be noted that our framework can easily accommodate any prior without major modifications. An additional factor $p(\boldsymbol{\theta})$ in Equation (3.7) directly leads to an additional summand $-\log(p(\boldsymbol{\theta}))$ in Equation (3.13). From a frequentist perspective, this could be interpreted as an additional regularizer, similarly to LASSO or ridge regression. Since all other summands in Equation (3.13) grow linearly with the amount of observations N and the prior contribution stays constant, the regularization term would eventually have minor influence in an asymptotic setting.

3.3.2 *Model Selection*

In practice, domain experts might not be able to provide one single true model. Instead, they might indicate a set of plausible models that they would like to test against the observed data. In this section we investigate this problem, known as model selection. For empirical evaluation, we use the Lotka-Volterra system as ground truth to simulate our empirical data. We then create four different candidate models via the following two additional ODEs

$$\dot{x}_1(t) = \quad \theta_1 x_1^2(t) + \theta_2 x_2(t), \tag{3.14}$$

$$\dot{x}_2(t) = -\theta_3 x_2(t). \tag{3.15}$$

Each model is indexed as $\mathcal{M}_{i,j}$, where $i, j \in \{0, 1\}$. Here, $i = 0$ indicates that the wrong equation (i.e. 3.14) is used to model the dynamics of the first state, while if $i = 1$ we provide the true parametric form in that specific candidate model. In a similar fashion, $j = 0$ indicates that the wrong equation (i.e. 3.15) is used to model the dynamics of the second state, otherwise $j = 1$. ODIN is run independently for each $\mathcal{M}_{i,j}$. Besides state and parameter estimates, we thus obtain final values for γ, which are presented in Table 3.2. For numerical stability, γ was lower bounded to 10^{-6} in all experiments. For the

correct model $\mathcal{M}_{1,1}$, γ settles at this lower bound, while it converges to a much larger value in case a wrong model is used. This justifies the intuitive interpretation of γ as a mean to account for model mismatch between the GP regressor and the ODE model. This last result proves that γ is indeed an efficient tool for identifying true parametric forms. Interestingly, this also works dimension-wise for the mixed models $\mathcal{M}_{0,1}$ and $\mathcal{M}_{1,0}$, even though the states x_1 and x_2 are coupled via wrong ODEs. This can be explained by the GP regressor prioritizing states $\boldsymbol{x}$ close to the observations $\boldsymbol{y}$. Indeed, while incorrect ODEs might deteriorate the accuracy of the state estimates with wrong regularization, their detrimental effects are limited by the observation-dependent partial objective, effectively decoupling the model mismatch across dimensions.

3.3.3 *Linear Scaling in State Dimension*

A key feature of gradient matching algorithms is the linear scaling in the state dimension K. Following Gorbach, Bauer, and Buhmann [GBB17], we demonstrate this for ODIN by using the Lorenz '96 system with $\theta = 8$, using 50 observations equally spaced over $t = [0, 5]$.

The Lorenz '96 system was originally introduced by Lorenz and Emanuel [LE98] for weather forecasting. The dimensionality $K > 3$ of this model can be chosen arbitrarily, with the k-th state being governed by the differential equation

$$\dot{x}_k = (x_{k+1} - x_{k-2})x_{k-1} - x_k + \theta, \tag{3.16}$$

where all indices should be read modulo K. The free choice of K makes it an ideal choice for scaling experiments.

The results are shown in Figure 3.8, including a linear regressor fitted to the means with least squares.

3.4 SLEIPNIR - SCALING TO BIGGER DATASETS

As shown in the previous section, ODIN scales linearly in the dimensionality of the states. Furthermore, since trajectories coming from different initial conditions are smoothed independently, ODIN also scales linearly in the number of trajectories. However, due to the $N_d \times N_d$ matrices of Equation (3.8) that need to be inverted, ODIN scales cubicly in the number of observations per trajectory. This is undesirable, especially for a smoothing-based algorithm, for which we expect that denser observations significantly improve its accuracy. Thus, until the end of this chapter, we will investigate,

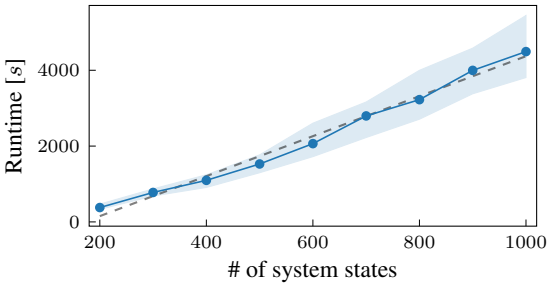


FIGURE 3.8: Run time for parameter inference on Lorenz '96 for different state dimension, with a (dashed) linear regressor fitted to the data. For each system size, we report the mean (dots) +- one standard deviation (shaded area) over 100 independent noise realizations.

develop and empirically study a novel way to consistently scale ODIN to a setting with many observations. Large parts of the material presented here was developed in a collaboration with colleagues. A preprint of that work can be found on arxiv [Ang+20]. Large parts of the presentation of that work and its plots have been adapted and reproduced to build the basis of the presentation here.

3.4.1 *Scaling standard GP Regression*

In the context of standard GP Regression, there exist several ideas on how to tackle the scaling problem. One family of approaches focuses on summarizing the data set with a fixed number of pseudo-observations, the so-called inducing points [QR05; SG06; Tit09]. Complementary to these methods, special structure in either the kernel function [Wil+14] or the input data [CSS08] can be exploited to speed up the necessary matrix-vector multiplications needed to perform GP regression. Wilson and Nickisch [WN15] combine these two ideas, creating an efficient approximation scheme linear in the number of observations. Independently, Sarkka, Solin, and Hartikainen [SSH13] propose a SDE-based reformulation and connect GP regression to Kalman filtering. Finally, there is a family of approaches approximating the kernel function via a finite dimensional scalar product of feature vectors. These features have been obtained using the Nyström method [WS01], Monte Carlo samples [RR08], sparse spectrum approximations [Laz+10], variational optimization [HDS+17] or a quadrature scheme [MK18]. Furthermore, Solin and Särkkä [SS] develop a deterministic feature expansion

based on the Fourier transform of the Laplace operator. While they are able to provide deterministic, non-asymptotic error bounds, their error decays linearly with the the size of the domain of the operator expansion L, which cannot grow faster than the number of features. This essentially means that their approximation error decays at best linearly.

3.4.2 *Scaling GP regression with derivatives*

In the context of Gaussian process regression with derivatives though, scalable methods seem to have received little attention [Eri+18], despite their practical relevance. While some approaches like inducing points could be applied by just using the same inducing points, it is not obvious how such changes would affect the approximation error of the resulting estimates. Similarly, an empirical extension of the approach of Solin and Särkkä [SS] to the derivative case is presented by Solin et al. [Sol+18], without quantifying the errors of the approximation scheme. For random Fourier features (RFF), Szabó and Sriperumbudur [SS19] present a rigorous theoretical analysis. However, due to the probabilistic nature of RFF, it is not possible to provide deterministic, non-asymptotic guarantees for any given data set of fixed size.

To obtain deterministic (i.e. hold with probability 1) and non-asymptotic (i.e. for data sets with a fixed number of observations and a fixed number of features) error bounds, we thus turn to quadratic Fourier features. Mutny and Krause [MK18] recently derived exponentially fast decaying bounds for standard GP regression. Building on their work, we derive approximations and bounds for derivative kernels as well. These bounds can then be used directly to quantify the absolute error of the predictive posterior mean and covariance of a GP with derivative observations, leading to deterministic, non-asymptotic error bounds. Besides guaranteeing users in safety-critical environments a guaranteed worst case performance, these bounds can also guide the choice of the number of features needed to obtain a desired accuracy.

3.4.3 *Accurately Approximating Derivative Kernels*

As outlined in the previous section, many feature approximations exist that all ultimately try to approximate the scalar product of a kernel $k(t_i, t_j) \approx \phi(t_i)^T \phi(t_j)$. The key idea is always the same: Any kernel represents a scalar product between two potentially infinite dimensional feature

vectors. However, if we allow for a small error, this could potentially be approximated by a finite-dimensional feature vector ϕ. Both feature approximations we will focus on in the following presentation, namely random Fourier features (RFF) introduced by Rahimi and Recht [RR08] and quadrature Fourier features (QFF) introduced by Mutny and Krause [MK18], have previously been studied intensively in the context of standard GP regression, while for RFF, there exist also analysis for derivative kernels [SS19]. As we shall see, they are both built on similar principles.

3.4.3.1 *Background - Fourier Features*

As before, we will introduce all concepts using a kernel with scalar inputs $k(t_i, t_j)$. However, it should be noted that this is by no means a necessary condition, and the concept generalizes nicely to higher dimensional inputs, albeit with an exponential penalty due to the curse of dimensionality. While this penalty is inherent to all kernel methods, it can be avoided by additive decomposition over groups of variables [MK18].

Both RFF and QFF are based on the following observation: For a stationary, scalar kernel, Bochner's theorem [see e.g. Rud91] guarantees the existence of a density $p(\omega)$ such that we can write

$$k(|t_i - t_j|) = \int_{-\infty}^{\infty} p(\omega) \begin{pmatrix} \cos(\omega t_i) \\ \sin(\omega t_i) \end{pmatrix}^T \begin{pmatrix} \cos(\omega t_j) \\ \sin(\omega t_j) \end{pmatrix} d\omega. \tag{3.17}$$

This equation can be interpreted as a scalar product of infinitely many features given by $\cos(\omega t_i)$ and $\sin(\omega t_i)$. To obtain a finite approximation, the aim is now to extract the M most important features.

To obtain the now famous random Fourier features, Rahimi and Recht [RR08] propose to use Monte Carlo integration. In one approximation, they obtain MC samples from $p(\omega)$ and for each sample add $[\sin(\omega t_i) \cos(\omega t_i)]$ to the feature vector $\phi(t_i)$. For further refinement, they observe that

$$\begin{pmatrix} \cos(\omega t_i) \\ \sin(\omega t_i) \end{pmatrix}^T \begin{pmatrix} \cos(\omega t_j) \\ \sin(\omega t_j) \end{pmatrix} = 2\mathbb{E}_b\{\cos(\omega t_i + b)\cos(\omega t_j + b)\}, \tag{3.18}$$

where $b \sim \text{Unif}([0, 2\pi])$. Thus, one can obtain a second approximation by sampling concurrently b and ω and for each sample adding $\sqrt{2}\cos(\omega t_i + b)$ to the feature vector $\phi(t_i)$. Accentuating the presence of the bias term b, we will refer to this approximation in the following as RFF-B, while we refer to the first one as RFF.

One main drawback of RFFs however is the fact that the intermediate sampling step makes it impossible to obtain any deterministic bounds on their approximation quality. Instead, all theoretical analysis has to be done either asymptotically or probabilistically.

Mutny and Krause [MK18] thus propose to approximate the integral of Equation (3.17) deterministically using Hermitian quadrature. In Hermitian quadrature, the locations at which to evaluate the integrand are not chosen via MC sampling. Instead, they are chosen to fulfill the following criterion: For any $f(x)$ being a polynomial of order $\leq 2m-1$, the Hermitian approximation Q_m of order m will accurately approximate

$$\int_{-\infty}^{\infty} w(x)f(x)dx = Q_m(f(\omega)) \coloneqq \sum_{i=1}^{m} W_i^m f(\omega_i^m), \tag{3.19}$$

where W_i^m and ω_i^m are pre-defined weights and abscissas, specifically designed for a pre-defined, non-negative weighting function $w(x)$ [cf. Hil87].

3.4.3.2 *Derivation - RBF Derivative Kernel*

Unfortunately, to apply the Hermitian formalism, the integral of Equation (3.17) has to be transformed to match the form of Equation (3.19). Thus, the resulting feature approximation depends on the functional form of the original kernel via $p(\omega)$, which is used to recover $w(x)$. In particular, this means that both derivation and error analysis have to be repeated for every new kernel. Thus, we restrict ourselves in this section to the RBF kernel, as required later in the applications presented in Section 3.6, keeping in mind that in principle, all analysis provided here could be extended to other stationary kernels. In the RBF case, we can use Gauss-Hermitian quadrature with $w(x) = e^{-x^2}$.

The RBF kernel is defined as $k(t_i, t_j) = k(t_i - t_j) = k(r) = \rho \exp(-\frac{r^2}{2l^2})$, where $r \coloneqq t_i - t_j$. It is parametrized by its hyperparameters ρ (variance) and l (lengthscale), that need to be learned in a pre-processing step. The RBF kernel is very widely used, also due to its strong smoothing properties. For sparsely observed systems, this smoothing can be too strong to capture all the finer dynamics of the system. For densely observed systems however, where the contribution of the prior is more and more disappearing, this is not an issue. Thus, it is a good choice for the applications we will show in Section 3.6.

For ease of readability, let us define $I(f(\omega)) \coloneqq \int_{-\infty}^{+\infty} e^{-\omega^2} f(\omega)d\omega$. To suppress an irrelevant, constant factor, we fix $\rho = \sqrt{\pi}$ for the theoretical

analysis. Using this notation, the RBF kernel can be written as $k(t_i, t_j) = I\left(\cos\left(\omega r\sqrt{2}/l\right)\right)$ [MK18]. To obtain the extension for derivative kernel, we differentiate this equality and use the linearity of the integral operator to obtain

$$\frac{\partial}{\partial t_i}k(t_i, t_j) = \frac{d}{dr}I\left(\cos\left(\omega r\sqrt{2}/l\right)\right) = I\left(-\sqrt{2}\omega\sin\left(\omega r\sqrt{2}/l\right)/l\right). \quad (3.20)$$

Similarly,

$$\frac{\partial}{\partial t_j}k(t_i, t_j) = I\left(\omega\frac{\sqrt{2}}{l}\sin\left(\omega r\frac{\sqrt{2}}{l}\right)\right) \quad (3.21)$$

and

$$\frac{\partial^2}{\partial t_i \partial t_j}k(t_i, t_j) = I\left(\omega^2\frac{2}{l^2}\cos\left(\omega r\frac{\sqrt{2}}{l}\right)\right). \quad (3.22)$$

In the spirit of QFF, these relations reduce the problem of approximating the kernel derivatives to approximating integrals. Thus, similar to the derivative-free case, we can now deploy Gauss-Hermite quadrature as in Equation (3.19) with $w(x) = e^{-x^2}$.

For a RBF kernel, Mutny and Krause [MK18] demonstrated that this boils down to approximating

$$\begin{aligned} k(t_i, t_j) &\approx Q_m(\cos(\omega r\sqrt{2}/l)) \\ &= \sum_{i=1}^{m} W_i^m \cos(\omega_i^m r\sqrt{2}/l) \\ &= \phi(t_i)^T\phi(t_j), \end{aligned} \quad (3.23)$$

where

$$\phi(x) \coloneqq \left[\sqrt{W_i^m}\cos(\omega_i^m t_i\sqrt{2}/l) \quad \sqrt{W_i^m}\sin(\omega_i^m t_i\sqrt{2}/l)\right]_{i=1\ldots m}^T. \quad (3.24)$$

Using trigonometric identities and a similar reasoning, we can show that

$$
\begin{aligned}
{}'k(t_i, t_j) &:= \frac{\partial}{\partial t_i} k(t_i, t_j) \quad \approx Q_m \left(-\sqrt{2}\omega \sin\left(\omega r \sqrt{2}/l\right)/l \right) \\
&= \phi(t_i)'^T \phi(t_j), \qquad (3.25) \\
k'(t_i, t_j) &:= \frac{\partial}{\partial t_j} k(t_i, t_j) \quad \approx Q_m \left(\sqrt{2}\omega \sin\left(\omega r \sqrt{2}/l\right)/l \right) \\
&= \phi(t_i)^T \phi'(t_j), \qquad (3.26) \\
k''(t_i, t_j) &:= \frac{\partial^2}{\partial t_i \partial t_j} k(t_i, t_j) \approx Q_m \left(\omega^2 2 \cos\left(\omega r \sqrt{2}/l\right)/l^2 \right) \\
&= \phi(t_i)'^T \phi'(t_j), \qquad (3.27)
\end{aligned}
$$

where

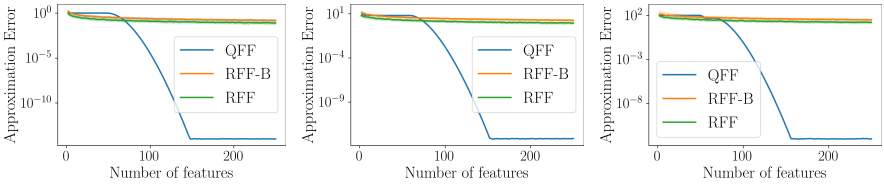
$$
\phi(t_i)' := \begin{bmatrix} -\sqrt{2W_i^m}\omega_i^m \sin\left(\omega_i^m t_i \sqrt{2}/l\right)/l \\ \sqrt{2W_i^m}\omega_i^m \cos\left(\omega_i^m t_i \sqrt{2}/l\right)/l \end{bmatrix}_{i=1\ldots m}. \qquad (3.28)
$$

As expected intuitively, the feature map of the derivative kernels looks like the derivative of the feature map of the original kernel, a fact that has heuristically been explored in similar context, e.g., by Solin and Särkkä [SS]. While this seems intuitively plausible, it is important to demonstrate that this map is then indeed optimal in the Gauss-Hermitian quadrature sense (cf. Equation (3.19)), a fact that is crucial for the later theoretical analysis. This crucial insight allows us to demonstrate that the feature expansions for the derivative kernels are efficient as well in the sense that the approximation error decays *exponentially* in the number of features. Only then can we claim that the expansion indeed provides accurate approximations of the derivative kernels even for a small number of *deterministically* chosen features.

3.4.3.3 *Theoretical Results - Exponentially Decaying Kernel Approximation Errors*

Since the approximation error of a GP posterior can vary hugely with the frequency content of the underlying function [see, e.g., WF22], we choose the kernel approximation error as a more objective metric. In the context of standard GP regression, Mutny and Krause [MK18] have bounded the kernel approximation error via the inequality

$$
|k(t_i, t_j) - \phi(t_i)^T \phi(t_j)| \leq E_m := E_m := \sqrt{\pi} \frac{1}{m^m} \left(\frac{e}{4l^2} \right)^m. \qquad (3.29)
$$



(a) $k(r)$ (b) $k'(r)$ (c) $k''(r)$

FIGURE 3.9: Maximum error of different feature expansions over $r \in [0, 1]$. For the stochastic RFF and RFF-B, median, 12.5% and 87.5% quantiles over 100 random samples are shown, but barely visible due to the exponential decay of the error of the QFF error. As predicted by our theoretical analysis, the error is a bit higher for the derivatives, but still decaying exponentially.

This bound is especially appealing since it is deterministic and decays exponentially in m. We prove that these favorable properties carry over to derivative kernels as well. The main results are summarized in Theorem 2, with a detailed proof given in Appendix A.2.3.

Theorem 2 *Let $k(t_i, t_j)$ be an RBF kernel and consider the Gauss-Hermite quadrature scheme of order m, defining $\phi(t_i)$ and $\phi(t_i)'$. Then, for $|r| = |t_i - t_j| \leq 1$, it holds that*

1. $|\frac{\partial}{\partial t_i} k(t_i, t_j) - \phi(t_i)'^T \phi(t_j)| \leq \frac{2e}{l^2} E_{m-2}$
2. $|\frac{\partial^2}{\partial t_i \partial t_j} k(t_i, t_j) - \phi(t_i)'^T \phi(t_j)'| \leq \frac{2e}{l^4} E_{m-3}$

where E_m is defined as in Equation (3.29)

In practical applications, most assumptions of the theorem are readily met, while $|r| < 1$ can always be achieved by scaling the domain and adapting the lengthscale l accordingly. In particular, if $|r|$ is large, scaling would decrease l proportionally, leading to a higher approximation error. However, it should be noted that this does not affect the exponential decay, since the scaling will just appear as a constant factor. While the error is slightly larger for the derivative approximations than for the kernel itself, it is important to note that we can still guarantee the same exponentially decaying behavior.

3.4.3.4 *Empirical Evaluation*

In Figure 3.9, we evaluate the maximum error of the feature approximations over an interval $r \in [0, 1]$. From Theorem 2, we would expect an exponential

decay in the approximation error for both the kernel and the first and second order derivatives. This exponential decay is clearly visible and continues until we hit machine precision. In Figure 3.9, l was chosen to be 0.1. However, as expected from our theoretical findings, this behavior is robust across different lengthscales. The corresponding plots can be found in the appendix (Appendix A.2.9). Figure 3.9 allows for the interesting observation that next to the exponential decay of QFF, the error of the RFF almost looks constant, even though it decays linearly on a non-logarithmic scale.

3.4.4 *Accurately Approximating Posteriors for GP Regression with Observed Derivatives*

Before investigating the more complex case of ODIN, we will focus in this section on the important application of Gaussian process regression with derivative observations (GPRD). As we shall see in this section, the exponentially fast decay of the kernel approximation error carries over nicely to this case.

3.4.4.1 *Background*

The generative model of GPRD, as introduced by Solak et al. [Sol+03], is shown in Figure 3.2. Unlike ODIN, in this case, we assume that we have access to the full set of state observations $\boldsymbol{y}$ and derivative observations $\boldsymbol{F}$. These vectors contain observations of the function itself and its derivatives at N_d distinct time points, corrupted by Gaussian noise with standard deviations σ and $\sqrt{\gamma}$. Given the GP prior, the goal of GPRD is to find estimates for $\mathbf{x}(t)$ and $\dot{\mathbf{x}}(t)$ given $\boldsymbol{y}$ and $\boldsymbol{F}$. Since both prior and observation model are Gaussian, the posteriors

$$p(x(\tau)|\boldsymbol{y}, \boldsymbol{F}, \phi, \gamma) = \mathcal{N}(x(\tau)|\mu(\tau), \alpha(\tau)) \tag{3.30}$$

and

$$p(x'(\tau)|\boldsymbol{y}, \boldsymbol{F}, \phi, \gamma) = \mathcal{N}(x'(\tau)|\mu'(\tau), \alpha'(\tau)). \tag{3.31}$$

are Gaussian as well and can be calculated in closed form.

To illustrate the computational challenges associated with this model, let's consider the calculation of the mean $\mu(T)$. Following e.g. [Sol+03]:

$$\mu(T) = \hat{\boldsymbol{k}}_\phi(\boldsymbol{t}, T)^T \hat{\boldsymbol{K}}_\phi^{-1} \left[\boldsymbol{y}^T, \boldsymbol{F}^T\right]^T, \tag{3.32}$$

where

$$\hat{\boldsymbol{K}}_{\boldsymbol{\phi}} := \begin{pmatrix} \boldsymbol{C}_{\boldsymbol{\phi}} & \boldsymbol{C}'_{\boldsymbol{\phi}} \\ {}'\boldsymbol{C}_{\boldsymbol{\phi}} & \boldsymbol{C}''_{\boldsymbol{\phi}} \end{pmatrix} + \begin{pmatrix} \sigma^2 \mathbb{I}_N & \mathbf{0} \\ \mathbf{0} & \gamma \mathbb{I}_N \end{pmatrix}. \tag{3.33}$$

Here, $\boldsymbol{C}_{\boldsymbol{\phi}}$, $\boldsymbol{C}'_{\boldsymbol{\phi}}$, ${}'\boldsymbol{C}_{\boldsymbol{\phi}}$ and $\boldsymbol{C}''_{\boldsymbol{\phi}}$ are all $N_d \times N_d$ matrices, describing the covariances between the states and derivatives, as given by k, ${}'k$, k' and k'' [cf. Section 2.3.1 and Sol+03]. To calculate the posterior, we would thus need to invert a $2N_d \times 2N_d$ matrix. Similar issues exist for the posterior derivative mean and the calculation of the variances as well, leading to an overall complexity of $\mathcal{O}(N_d^3)$.

3.4.5 *Derivation*

However, similar to scaling standard GP regression, $\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}$ can be decomposed as $\hat{\boldsymbol{K}}_{\boldsymbol{\phi}} \approx \hat{\boldsymbol{\Phi}}^T \hat{\boldsymbol{\Phi}} + \boldsymbol{\Lambda}$. Here, $\hat{\boldsymbol{\Phi}} \in \mathbb{R}^{2M \times N_d}$ is the feature matrix we obtain when stacking the N_d 2M-dimensional feature vectors obtained by concatenating $\boldsymbol{\phi}(t_i)$ and $\boldsymbol{\phi}'(t_i)$ for t_i. Also $\boldsymbol{\Lambda}$ is the diagonal matrix we obtain due to the noise variance terms γ and σ^2. Using the matrix inversion lemma, we can write

$$\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1} \approx (\hat{\boldsymbol{\Phi}}^T \hat{\boldsymbol{\Phi}} + \boldsymbol{\Lambda})^{-1} = \boldsymbol{\Lambda}^{-1} - \boldsymbol{\Lambda}^{-1} \hat{\boldsymbol{\Phi}}^T (\boldsymbol{I} + \hat{\boldsymbol{\Phi}} \boldsymbol{\Lambda}^{-1} \hat{\boldsymbol{\Phi}}^T)^{-1} \hat{\boldsymbol{\Phi}} \boldsymbol{\Lambda}^{-1}. \tag{3.34}$$

This trick allows us to compute the inverse of $\hat{\boldsymbol{K}}_{\boldsymbol{\phi}} \in \mathbb{R}^{2N_d \times 2N_d}$ by calculating the inverse of the (much) smaller $(\boldsymbol{I} + \hat{\boldsymbol{\Phi}} \boldsymbol{\Lambda}^{-1} \hat{\boldsymbol{\Phi}}^T) \in \mathbb{R}^{2M \times 2M}$. Thus, it should be clear that the original complexity of $\mathcal{O}(N_d^3)$ has been successfully reduced to $\mathcal{O}(N_d M^2 + M^3)$.

Note that this is only possible since the features needed to approximate k and k'' are the same ones we need to approximate k'. In that sense, the features derived in the previous section are not only optimal in a Hermitian quadrature sense, but also ideal for this feature approximation.

3.4.5.1 *Theoretical Results*

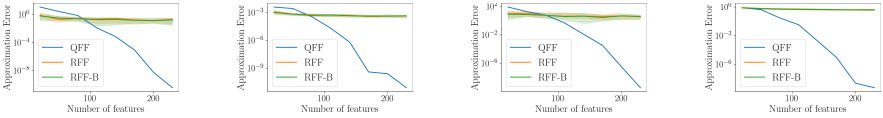
As we shall see, the exponential error decay carries over nicely to the case of GPRD. Define $e_{\tilde{\mu}}$, $e_{\tilde{\alpha}}$, $e_{\tilde{\mu}'}$ and $e_{\tilde{\alpha}'}$ as the absolute error between the feature approximations and the corresponding accurate quantities of the means and covariances of Equations (3.30) and (3.31). For each $\tau \in \mathbb{R}$, define $e_{\text{tot}} := \max\{e_{\tilde{\mu}}, e_{\tilde{\alpha}}, e_{\tilde{\mu}'}, e_{\tilde{\alpha}'}\}$ as the maximum of these four errors. We can now show the exponential relation between feature approximation order m

and the corresponding approximation error, as summarized in Theorem 3, with proof in Appendix A.2.3.

Theorem 3 *Let us consider an RBF kernel with hyperparameters* (ρ, l) *and domain* $[0,1]$. *Define* $c := \min(\gamma, \sigma^2)$ *and* $R := \max(||\boldsymbol{y}||_\infty, ||\boldsymbol{F}||_\infty)$. *Let* $C > 0$. *Let us consider a QFF approximation scheme of order* m *with*
$m \geq 3 + \max\left(\frac{e}{2l^2}, \log\left(\frac{270N^2\rho^3R}{l^8c^2C}\right)\right)$.
Then, it holds for all $\tau \in [0,1]$ *that* $e_{tot} \leq C$.

From this theorem, we can also observe the following fact: If we decrease the acceptable worst case performance C, we clearly need more features. However, due to the logarithm in the theorem, an *exponential* decay in C only leads to *linear* growth in m, all other things being equal.

3.4.5.2 *Empirical Evaluation*



(a) μ_0 (b) α_0 (c) μ_0' (d) α_0'

FIGURE 3.10: Approximation error of the feature approximations compared to the accurate GP, evaluated at $t = 0.8$ for the first state of the Lorenz system, with 1000 observations and an SNR of 100. Median, 12.5% and 87.5% quantiles over 10 independent noise realizations are shown.

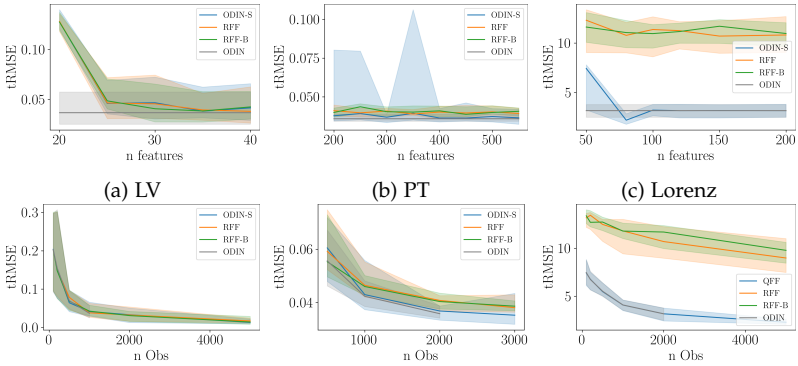
Wherever applicable, we will use three standard benchmark systems previously introduced in Sections 2.7.1 and 3.3.3. For all experiments, we created our data-set using numerical integration of the ground truth parameters. We then added 25 different noise realizations to obtain 25 different data sets. This allows us to quantify robustness w.r.t. noise by showing median as well as 20% and 80% quantiles over these noise realizations for each experiment. In all experiments, we trained γ and learned the kernel hyperparameters from the data using the scalable approximations described in the previous section. For all algorithms, we keep RFF and RFF-B feature expansions as comparison to our QFF-based approximation.

For GPRD, we show the exponential decay of all error quantities described by Theorem 3 for the Lorenz system in Figure 3.10. It is clear that the

	LV	PT	Lorenz
ODIN	973 ± 42.5	17900 ± 668	10700 ± 466
ODIN-S	**1.24 ± 0.243**	**133 ± 12.8**	**13.2 ± 0.566**

TABLE 3.3: Run time per iteration in miliseconds. Median ± on standard deviation shown over 15 different iterations.

predicted exponential decay also holds in practice. While we only show one state of one experiment, this behavior is consistent across experiments and noise settings, as can be seen by looking at the additional plots presented in Appendix A.2.9.



(a) LV (b) PT (c) Lorenz

FIGURE 3.11: Comparing the tRMSE of the inferred ODE parameters. Top row: The results obtained by ODIN-S converge to the accurate results when increasing the number of features. Bottom row: Only ODIN-S is accurate enough to not affect the learning curves.

3.5 ODIN WITH SLEIPNIR - RISK CONSISTENT SCALING

The first computational challenge for ODIN lies in calculating the risk term of Equation (3.13), which can be written slightly more verbose as

$$\begin{aligned}\mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}) = &\boldsymbol{x}^T \boldsymbol{C}_{\boldsymbol{\phi}}^{-1} \boldsymbol{x} \\ &+ (\boldsymbol{x} - \boldsymbol{y})^T \sigma^{-2} (\boldsymbol{x} - \boldsymbol{y}) \\ &+ (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{D}\boldsymbol{x})^T (\boldsymbol{A} + \gamma \boldsymbol{I})^{-1} (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{D}\boldsymbol{x}).\end{aligned} \tag{3.35}$$

Clearly, ODIN scales similarly to standard GPRD, cubically in the number of observations, due to the inversions of $\boldsymbol{C}_\phi$ and $\boldsymbol{A}$. Also, if γ is inferred from the data as well (in this section referred by learning gamma), $\mathcal{R}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})$ is extended by the additional summand $\log\det(\boldsymbol{A} + \gamma \boldsymbol{I})$. This term represents the contribution of the determinant term of the conditional $p(\boldsymbol{F}|\boldsymbol{x}, \boldsymbol{y}, \phi, \gamma)$ and also scales cubically. Furthermore, when inferring the hyper-parameters ϕ and σ from data via empirical Bayes. Here, ODIN maximizes the marginal likelihood $p(\boldsymbol{y}|\phi, \sigma)$. Usually, this is done via the log likelihood, whose calculation scales cubically as well. So in essence, there are three main challenges when scaling up ODIN to large datasets, where the latter two have already intensely been studied, e.g. by Rahimi and Recht [RR08] or Mutny and Krause [MK18]. Thus, we focus in this section on deriving the appropriate approximations for the risk terms.

3.5.1 *Derivation*

Unlike the matrices needed for GPRD, neither $\boldsymbol{D}$ nor $\boldsymbol{A}$ allow for a direct feature expansion. However, after inserting the appropriate feature expansions and applying the matrix inversion Lemma multiple times, we can obtain

$$\boldsymbol{z}^T(\boldsymbol{A} + \gamma \boldsymbol{I})^{-1}\boldsymbol{z} \approx \frac{1}{\gamma}\boldsymbol{z}^T(\mathbb{I} - \boldsymbol{\Phi}'^T(\boldsymbol{\Phi}'\boldsymbol{\Phi}'^T + \frac{\gamma}{\lambda}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \gamma\mathbb{I})^{-1}\boldsymbol{\Phi}')\boldsymbol{z},$$

where

$$\boldsymbol{z} \coloneqq \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{D}\boldsymbol{x} \approx \boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{x}) - \boldsymbol{\Phi}'^T(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\mathbb{I})^{-1}\boldsymbol{\Phi}\boldsymbol{x}.$$

Combining these approximations with the ones introduced in the previous section, we obtain a computationally efficient scheme for calculating all objective functions in ODIN. Furthermore, both the additional term $\log\det(\boldsymbol{A} + \gamma \boldsymbol{I})$ obtained when learning γ and learning ϕ and σ can be scaled similarly. Overall, this means that the original complexity of $\mathcal{O}(N_d^3)$ has been successfully reduced to $\mathcal{O}(N_d M^2 + M^3)$. For $M < N_d$, this is significantly accelerating ODIN, which is why we name our approximation scheme SLEIPNIR, as a reference to norse mythology. The resulting algorithm will be referred to as ODIN-S.

THEORETICAL RESULTS When applying SLEIPNIR to ODIN for a fixed feature vector length M, we will always create a small approximation error

in the objective function. However, the deterministic nature of the QFF approximation still allows us to choose the number of features in a way such that we can guarantee it to be smaller than a pre-chosen threshold. This result is summarized in Theorem 4 and proven in Appendix A.2.7.

Theorem 4 *Let $\mathcal{R}$ be the* ODIN*-objective as defined in Equation* (3.35) *and let $\tilde{\mathcal{R}}$ denote its counterpart obtained by approximating the matrices $\boldsymbol{C}_{\boldsymbol{\phi}}$, $\boldsymbol{A}$ and $\boldsymbol{D}$ with a QFF scheme of order m. Assume the parameters λ, γ, $\boldsymbol{\phi} = (\rho, l)$ and N_d to be fixed. Suppose $N_d \geq 60$ and let $1 > \epsilon > 0$. Then, for any $\boldsymbol{x}$ and $\boldsymbol{\theta}$,*

$$m \geq 10 + \max\left\{\frac{e}{2l^2}, \log_2\left(\frac{\rho^2 N_d^3}{\lambda^2 \gamma l^4 \epsilon}\right)\right\} \tag{3.36}$$

implies

$$\frac{|R_{\lambda\gamma\phi}(x,\theta) - \tilde{R}_{\lambda\gamma\phi}(x,\theta)|}{R_{\lambda\gamma\phi}(x,\theta)} \leq \epsilon \tag{3.37}$$

Similar to GPRD, the threshold ϵ appears inside the logarithm. Thus, an *exponential* decrease of the allowed error only requires a *linear* increase in the number of features.The bound on N_d has been chosen to reduce the number of terms, but it is not necessary. The bound on ϵ is equivalent to stating that an approximation scheme with more than 100% relative error is of little interest.

3.6 SCALING EXPERIMENTS

In Figure 3.11, we compare the performance of ODIN-S against the original ODIN as well as ODIN augmented with RFF and RFF-B on the three standard benchmark systems. In the top row, we keep the total number of observations fixed to 1000 for LV and 2000 for PT and Lorenz, while varying the length of the feature vector. In the bottom row, we keep the number of features fixed to 40 for LV, 300 for PT and 150 for Lorenz. All the data was created using observation noise with $\sigma^2 = 0.1$ for LV, $\sigma^2 = 0.01$ for PT and a signal-to-noise ratio of 5 for Lorenz. Due to computational restrictions, it was not possible to evaluate accurate ODIN beyond what is shown in the plots. To demonstrate the robustness of the evaluation, different noise, feature and observation settings are investigated in Appendix A.2.9.

As predicted by our theoretical analysis, the trajectory RMSE of ODIN-S eventually converges to the approximation-free ODIN in both median and quantiles if we increase the number of QFFs. Also, it is interesting

to observe that the learning curves of the MCMC-based RFF and RFF-B are significantly worse than the true learning curves, especially for the more involved PT and Lorenz systems. This seems to suggest that a bad approximation can seriously hurt learning performance, further illustrating the importance of provably accurate approximations.

In Table 3.3, we compare the run time per iteration of ODIN with ODIN-S. Since the run time only depends on the number of features and not on how the features were obtained, we omitted RFF and RFF-B. The run time was evaluated using the same number of observation and number of feature combinations as in Figure 3.11. While the table only shows the run time reduction, the theoretically expected linear scaling is demonstrated in additional plots in the appendix.

In a final experiment, we show that ODIN-S is able to scale to realistic data sets. For this, we introduce a 12-dimensional ODE system representing the dynamics of a 6DOF quadrocopter. We observe the system under Gaussian noise with SNR=10 over the time interval $t = [0, 15]$. We assume a sampling frequency of 1kHz, leading to 15'000 observations. We then run ODIN-S on a standard laptop (Lenovo Carbon X1) and obtain results in roughly 80min. Up to our knowledge, this is the first time that a system of such dimensions has been solved with a Gaussian process based parameter inference scheme, clearly demonstrating the power of our framework. The resulting trajectories are shown in Figure 3.12, including example observation points to visualize the noise level. The estimates of ODIN-S are so good that the ground truth is almost indistinguishable.

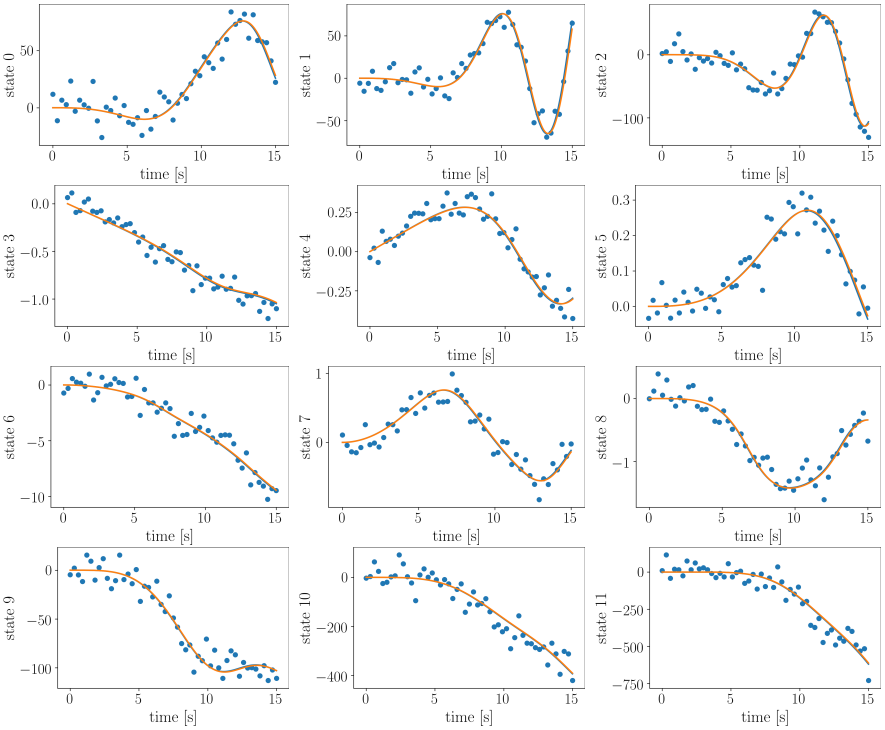


FIGURE 3.12: State trajectories obtained by integrating the parameters inferred by ODIN-S (orange). The blue line represents the ground truth, while the blue dots show every 300-th observation for a signal-to-noise ratio of 10.

4

THE PRACTITIONER: DISTRIBUTIONAL GRADIENT MATCHING

In all dynamics models studied in this thesis thus far, we assumed that the dynamics model was known and the number of parameters quite limited. In particular, all algorithms thus far were evaluated on single trajectory datasets. While their frameworks can in principle be extended to the multi-trajectory case without theoretical problems, this is rarely done. In part, this is to inherent computational limitations. In part, this is to keep in line with the current gradient matching literature, which almost exclusively studies the one-trajectory case. In part, this is also due to the fact that the parametric form of the established benchmarks are quite restrictive. For gradient matching to work in its established form, trajectories need to be somewhat densely observed such that the smoother can pick up important characteristics of the underlying dynamics. And for such densely observed dynamics, one trajectory is already enough to learn decent dynamics for the systems considered thus far.

In practice however, we might be fundamentally limited in the frequency at which we can take measurements, so densely observing trajectories might not be an option. Furthermore, we might be interested in studying a system's behavior in different regions of the state space, i.e. starting from different initial conditions, especially in the context of system identification [Lju98]. This could result in data sets with many observed trajectories, but with fewer observations on each trajectory than studied thus far. A naive extension of FGPGM or ODIN to this case would involve learning a smoother for each trajectory independently. Even for simple smoothers with few hyperparameters, this could quickly lead to an explosion of learnable parameters. Thus, we would be forced to adapt the complexity of each smoother to the number of observations available per trajectory, which might fundamentally limit our capacity to accurately learn the underlying characteristics of each trajectory.

In this chapter, we investigate a possible solution to this problem. Instead of deploying independent smoothers for each trajectory, we propose to learn a joint smoother over all trajectories. This smoother is then regularized by a neural ordinary differential equation (neural ODE), to guarantee that its predictions actually follow some underlying, Markovian dynamical system.

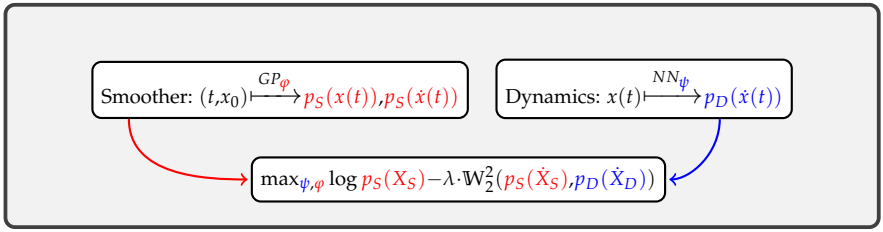


FIGURE 4.1: High-level depiction of DGM.

The first parts of this chapter up to the case study were adapted from the joint work with Treven et al. [Tre+21]. This work was published as a conference publication and parts of it are reused with permission.

A high-level depiction of our algorithm, called distributional gradient matching (DGM), is shown in Figure 4.1. In principle, DGM jointly learns a smoother (S) and a dynamics model (D). The smoother model, chosen to be a Gaussian process, maps an initial condition x_0 and a time t to the state distribution $p_S(x(t))$ and state derivatives distribution $p_S(\dot{x}(t))$ reached at that time. The dynamics model, chosen to be a neural network, represents an ODE that maps states $x(t)$ to the derivative distribution $p_D(\dot{x}(t))$. Both models are evaluated at some training times and all its output distributions collected in the random variables X_S, $\dot{X}_S$ and $\dot{X}_D$. The parameters of these models are then jointly trained using a Wasserstein-distance-based objective directly on the level of distributions.

4.1 BACKGROUND

4.1.1 *Data*

As in the previous chapters, we again consider a data set as described in Section 1.1. For the first time in this thesis however, we explicitly consider the case of multiple trajectories. To introduce DGM properly, a simplification to the one-trajectory case as done in the previous chapters for brevity of notation is not helpful here. Whenever we need an explicit reference to a specific trajectory, we will use a subindex $m \in \{1, \dots, M\}$, to indicate the index of the numbered M trajectories. Since there is no need for any restrictions, we let the number of observations $(N_m)_{m \in \{1,\dots,M\}}$ vary from trajectory to trajectory. Thus, a trajectory m is described by its initial condi-

tion $\boldsymbol{x}_m(0)$, and the observations $\boldsymbol{y}_m \coloneqq [\boldsymbol{x}_m(t_{n,m}) + \boldsymbol{\epsilon}_{n,m}]_{n\in\{1,\dots,N_m\}}$ at times $\boldsymbol{t}_m \coloneqq [t_{n,m}]_{n\in\{1,\dots,N_m\}}$, where the additive observation noise $\boldsymbol{\epsilon}_{n,m}$ is assumed to be drawn i.i.d. from a zero mean Gaussian, whose covariance is given by $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} \coloneqq \operatorname{diag}(\sigma_1^2, \dots, \sigma_K^2)$. We denote by $\mathcal{D}$ the dataset, consisting of M initial conditions $\boldsymbol{x}_m(0)$, observation times $\boldsymbol{t}_m$, and observations $\boldsymbol{y}_m$, in line with the definitions introduced in Section 1.1 and the cases studied thus far in this thesis.

4.1.2 *Problem*

To model the unknown dynamical system, we choose again a deterministic ODE

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{4.1}$$

where we do not further restrict the parametric form of $\boldsymbol{f}$, except for requiring the ground truth to be Lipschitz so that unique system evolutions exist for each initial condition. Thus, depending on the amount of expert knowledge, the parametrization of $\boldsymbol{f}$ can follow a white-box, gray-box, or black-box methodology [Boho6]. In any case, the parametric form of $\boldsymbol{f}$ is fixed a priori (e.g., a neural network), and the first key challenge is to infer a reasonable distribution over the parameters $\boldsymbol{\theta}$, conditioned on the data $\mathcal{D}$ (i.e. Bayesian inference, as introduced in Section 1.1.1).

For later tasks, we are particularly interested in the *predictive posterior state distribution*

$$p(\boldsymbol{x}_{\text{new}}(\boldsymbol{t}_{\text{new}}) | \mathcal{D}, \boldsymbol{t}_{\text{new}}, \boldsymbol{x}_{\text{new}}(0)), \tag{4.2}$$

i.e., the posterior distribution of the states starting from a potentially unseen initial condition $\boldsymbol{x}_{\text{new}}(0)$ and evaluated at times $\boldsymbol{t}_{\text{new}}$. This posterior would then be used by the downstream or prediction tasks described at the beginning of this chapter.

4.1.3 *Motivation*

As mentioned in Section 1.1, standard Bayesian inference requires many integration steps. The same holds for the practically important prediction as described in Equation (4.2). Especially when we model $\boldsymbol{f}$ with a neural network, this can be a huge burden, both numerically and computationally [Kel+20].

As an alternative approach, we can approximate the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ with variational inference [Bis06]. However, we run into similar bottlenecks.

While optimizing the variational objective, e.g., the ELBO, many integration steps are necessary to evaluate the unnormalized posterior. Also, at inference time, to obtain a distribution over state $\hat{\boldsymbol{x}}_s(t)$, we still need to integrate $\boldsymbol{f}$ several times. Furthermore, Dandekar et al. [Dan+21] report poor forecasting performance by the variational approach.

4.2 DISTRIBUTIONAL GRADIENT MATCHING

In both the Monte Carlo sampling-based and variational approaches, all information about the dynamical system is stored in the estimates of the system parameters $\hat{\boldsymbol{\theta}}$. This makes these approaches rather cumbersome: Both for obtaining estimates of $\hat{\boldsymbol{\theta}}$ and for obtaining the predictive posterior over states, once $\hat{\boldsymbol{\theta}}$ is found, we need multiple rounds of numerically integrating a potentially complicated (neural) differential equation. We thus have identified two bottlenecks limiting the performance and applicability of these algorithms: namely, numerical integration of $\boldsymbol{f}$ and inference of the system parameters $\boldsymbol{\theta}$. In our proposed algorithm, we *avoid both* of these bottlenecks by directly working with the posterior distribution in the state space.

To this end, we introduce a probabilistic, differentiable *smoother model*, that directly maps a tuple $(t, \boldsymbol{x}(0))$ consisting of a time point t and an initial condition $\boldsymbol{x}(0))$ as input and maps it to the corresponding distribution over $\boldsymbol{x}(t)$. Thus, the smoother directly replaces the costly, numerical integration steps, needed, e.g., to evaluate Equation (4.2).

Albeit computationally attractive, this approach has one serious drawback. Since the smoother no longer explicitly integrates differential equations, there is no guarantee that the obtained smoother model follows any vector field. Thus, the smoother model is strictly more general than the systems described by deterministic ODEs as defined in Section 1.1. Unlike ODEs, it is able to capture mappings whose underlying functions violate, e.g., Lipschitz or Markovianity properties, which is clearly not desirable. To address this issue, we introduce a regularization term, $\mathcal{L}_{\text{dynamics}}$, which ensures that a trajectory predicted by the smoother is encouraged to follow some underlying ODE system. The smoother is then trained with the multi-objective loss function

$$\mathcal{L} := \mathcal{L}_{\text{data}} + \lambda \cdot \mathcal{L}_{\text{dynamics}}, \tag{4.3}$$

where, $\mathcal{L}_{\text{data}}$ is a smoother-dependent loss function that ensures a sufficiently accurate data fit, and λ is a trade-off parameter.

4.2.1 *Regularization by Matching Distributions over Gradients*

To ultimately define $\mathcal{L}_{\text{dynamics}}$, first choose a parametric *dynamics model* similar to $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$ in Equation (1.1), that maps states to their derivatives. Second, define a set of *supporting points* $\mathcal{T}$ with the corresponding *supporting gradients* $\dot{\mathcal{X}}$ as

$$\mathcal{T} \coloneqq \left\{ \left(t_{\text{supp},l}, \boldsymbol{x}_{\text{supp},l}(0) \right)_{l \in \{1 \dots N_{\text{supp}}\}} \right\},$$
$$\dot{\mathcal{X}} \coloneqq \left\{ \left(\dot{\boldsymbol{x}}_{\text{supp},l} \right)_{l \in \{1 \dots N_{\text{supp}}\}} \right\}.$$

Here, the l-th element represents the event that the dynamical system's derivative at time $t_{\text{supp},l}$ is $\dot{\boldsymbol{x}}_{\text{supp},l}$, after being initialized at time 0 at initial condition $\boldsymbol{x}_{\text{supp},l}(0)$.

Given both the smoother and the dynamics model, we have now two different ways to calculate distributions over $\dot{\mathcal{X}}$ given some data $\mathcal{D}$ and supporting points $\mathcal{T}$. First, we can directly leverage the differentiability and global nature of our smoother model to extract a distribution $p_S(\dot{\mathcal{X}}|\mathcal{D}, \mathcal{T})$ from the smoother model. Second, we can first use the smoother to obtain state estimates and then plug these state estimates into the dynamics model, to obtain a second distribution $p_D(\dot{\mathcal{X}}|\mathcal{D}, \mathcal{T})$. Clearly, if the solution proposed by the smoother follows the dynamics, these two distributions should match. Thus, we can regularize the smoother to follow the solution of Equation (1.1) by defining $\mathcal{L}_{\text{dynamics}}$ to encode the *distance* between $p_D(\dot{\mathcal{X}}|\mathcal{D}, \mathcal{T})$ and $p_S(\dot{\mathcal{X}}|\mathcal{D}, \mathcal{T})$ to be small in some metric. By minimizing the overall loss, we thus match the distributions over the gradients of the smoother and the dynamics model.

4.2.2 *Smoothing jointly over Trajectories with Deep Gaussian Processes*

The core of DGM is formed by a smoother model. In principle, the posterior state distribution of Equation (4.2) could be modeled by any Bayesian regression technique. However, calculating $p_S(\dot{\mathcal{X}}|\mathcal{D}, \mathcal{T})$ is generally more involved. Here, the key challenge is evaluating this posterior, which is already computationally challenging, e.g., for simple Bayesian neural networks. For Gaussian processes, however, this becomes straightforward, since derivatives of GPs remain GPs [Sol+03]. Thus, DGM uses a GP smoother. For scalability and simplicity, we keep K different, independent smoothers,

one for each state dimension. However, if computational complexity is not a concern, our approach generalizes directly to multi-output Gaussian processes. Below, we focus on the one-dimensional case, for clarity of exposition. For notational compactness, all vectors with a superscript should be interpreted as vectors over time in this subsection. For example, the vector $\boldsymbol{x}^{(k)}$ consists of all the k-th elements of the state vectors $\boldsymbol{x}(t_{n,m}), n \in \{1, \ldots, N_m\}, m \in \{1, \ldots, M\}$.

We define a Gaussian process with a differentiable mean function

$$\mu(\boldsymbol{x}_m(0), t_{n,m})$$

as well as a differentiable and positive-definite kernel function

$$\mathcal{K}_{\text{RBF}}(\boldsymbol{\phi}(\boldsymbol{x}_m(0), t_{n,m}), \boldsymbol{\phi}(\boldsymbol{x}_{m'}(0), t_{n',m'}).$$

Here, the kernel is given by the composition of a standard ARD-RBF kernel [Raso4b] and a differentiable feature extractor ϕ parametrized by a deep neural network, as introduced by Wilson et al. [Wil+16]. Following Solak et al. [Sol+03], given fixed $\boldsymbol{x}_{\text{supp}}$, we can now calculate the joint density of $(\dot{\boldsymbol{x}}_{\text{supp}}^{(k)}, \boldsymbol{y}^{(k)})$ for each state dimension k. Concatenating vectors accordingly across time and trajectories, let

$$\begin{aligned}
\boldsymbol{\mu}^{(k)} &:= \mu^{(k)}\left(\boldsymbol{x}(0), \boldsymbol{t}\right), \quad \dot{\boldsymbol{\mu}}^{(k)} := \frac{\partial}{\partial t}\mu^{(k)}\left(\boldsymbol{x}_{\text{supp}}(0), \boldsymbol{t}_{\text{supp}}\right), \\
\boldsymbol{z}^{(k)} &:= \boldsymbol{\phi}^{(k)}(\boldsymbol{x}(0), \boldsymbol{t}), \qquad \boldsymbol{z}_{\text{supp}}^{(k)} := \boldsymbol{\phi}^{(k)}(\boldsymbol{x}_{\text{supp}}(0), \boldsymbol{t}_{\text{supp}}), \\
\boldsymbol{\mathcal{K}}^{(k)} &:= \mathcal{K}_{\text{RBF}}^{(k)}(\boldsymbol{z}^{(k)}, \boldsymbol{z}^{(k)}), \quad \dot{\boldsymbol{\mathcal{K}}}^{(k)} := \frac{\partial}{\partial t_1}\mathcal{K}_{\text{RBF}}^{(k)}(\boldsymbol{z}_{\text{supp}}^{(k)}, \boldsymbol{z}^{(k)}), \\
\ddot{\boldsymbol{\mathcal{K}}}^{(k)} &:= \frac{\partial^2}{\partial t_1 \partial t_2}\mathcal{K}_{\text{RBF}}^{(k)}(\boldsymbol{z}_{\text{supp}}^{(k)}, \boldsymbol{z}_{\text{supp}}^{(k)}).
\end{aligned}$$

Then the joint density of $(\dot{\boldsymbol{x}}_{\text{supp}}^{(k)}, \boldsymbol{y}^{(k)})$ can be written as

$$\begin{pmatrix} \dot{\boldsymbol{x}}_{\text{supp}}^{(k)} \\ \boldsymbol{y}^{(k)} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \dot{\boldsymbol{\mu}}^{(k)} \\ \boldsymbol{\mu}^{(k)} \end{pmatrix}, \begin{pmatrix} \ddot{\boldsymbol{\mathcal{K}}}^{(k)} & \dot{\boldsymbol{\mathcal{K}}}^{(k)} \\ (\dot{\boldsymbol{\mathcal{K}}}^{(k)})^\top & \boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2 \boldsymbol{I} \end{pmatrix}\right). \tag{4.4}$$

Here we denote by $\frac{\partial}{\partial t_1}$ the partial derivative with respect to time in the first coordinate, by $\frac{\partial}{\partial t_2}$ the partial derivative with respect to time in the second coordinate, and with σ_k^2 the corresponding noise variance of $\boldsymbol{\Sigma}_{\text{obs}}$.

Since the conditionals of a joint Gaussian random variable are again Gaussian distributed, p_S is again Gaussian, i.e.,

$$p_S(\dot{\mathcal{X}}_k|\mathcal{D},\mathcal{T}) = \mathcal{N}\left(\dot{\boldsymbol{x}}^{(k)}_{\text{supp}}|\boldsymbol{\mu}_S,\boldsymbol{\Sigma}_S\right)$$

with

$$\begin{aligned}\boldsymbol{\mu}_S &\coloneqq \dot{\boldsymbol{\mu}}^{(k)} + \dot{\boldsymbol{\mathcal{K}}}^{(k)}(\boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2\boldsymbol{I})^{-1}\left(\boldsymbol{y}^{(k)} - \boldsymbol{\mu}^{(k)}\right),\\ \boldsymbol{\Sigma}_S &\coloneqq \ddot{\boldsymbol{\mathcal{K}}}^{(k)} - \dot{\boldsymbol{\mathcal{K}}}^{(k)}(\boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2\boldsymbol{I})^{-1}(\dot{\boldsymbol{\mathcal{K}}}^{(k)})^\top.\end{aligned} \tag{4.5}$$

Here, the index k is used to highlight that this is just the distribution for one state dimension. To obtain the final $p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$, we take the product over all state dimensions k.

To fit our model to the data, we minimize the negative marginal log likelihood of our observations, neglecting purely additive terms [Rasoqb], i.e.,

$$\begin{aligned}\mathcal{L}_{\text{data}} \coloneqq \frac{1}{2}\sum_{k=1}^{K}&\left(\boldsymbol{y}^{(k)} - \boldsymbol{\mu}^{(k)}\right)^\top\left(\boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2\boldsymbol{I}\right)^{-1}\left(\boldsymbol{y}^{(k)} - \boldsymbol{\mu}^{(k)}\right)\\ &+ \operatorname{logdet}\left(\boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2\boldsymbol{I}\right).\end{aligned} \tag{4.6}$$

Furthermore, the predictive posterior for a new point $x^{(k)}_{\text{test}}$ given time t_{test} and initial condition $x^{(k)}_{\text{test}}(0)$ has the closed form

$$p_S(x^{(k)}_{\text{test}}|\mathcal{D}_k, t_{\text{test}}, \boldsymbol{x}_{\text{test}}) = \mathcal{N}\left(x^{(k)}_{\text{test}}\middle|\mu^{(k)}_{\text{post}}, \sigma^2_{\text{post},k}\right), \tag{4.7}$$

where

$$\begin{aligned}\mu^{(k)}_{\text{post}} &= \mu^{(k)}(\boldsymbol{x}_{\text{test}}(0), t_{\text{test}})\\ &\quad + \mathcal{K}^{(k)}_{\text{RBF}}(\boldsymbol{z}^{(k)}_{\text{test}}, \boldsymbol{z}^{(k)})^\top(\boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2\boldsymbol{I})^{-1}\left(\boldsymbol{y}^{(k)} - \boldsymbol{\mu}^{(k)}\right)\end{aligned} \tag{4.8}$$

and

$$\begin{aligned}\sigma^2_{\text{post},k} &= \mathcal{K}^{(k)}_{\text{RBF}}(\boldsymbol{z}_{\text{test}}, \boldsymbol{z}_{\text{test}})\\ &\quad - \mathcal{K}^{(k)}_{\text{RBF}}(\boldsymbol{z}^{(k)}_{\text{test}}, \boldsymbol{z}^{(k)})^\top(\boldsymbol{\mathcal{K}}^{(k)} + \sigma_k^2\boldsymbol{I})^{-1}\mathcal{K}^{(k)}_{\text{RBF}}(\boldsymbol{z}^{(k)}_{\text{test}}, \boldsymbol{z}^{(k)}).\end{aligned} \tag{4.9}$$

Please note that the design choices in this section are by no means absolute. All we need from the smoother model is a mechanism to extract a probabilistic posterior. In particular, it could be interesting to explore

alternative smoother models. In the context of model selection, Pfister, Bauer, and Peters [PBP18] successfully deployed a spline-based approach. As a more Bayesian alternative, Cutajar et al. [Cut+17] introduce a scalable alternative in the context of GPs, although their definition of deep GP should not be confused with the deep GPs used in this thesis. For the sake of simplicity, we leave the investigation of such approaches to future work though.

4.2.3 *Representing Uncertainty in the Dynamics Model via the Reparametrization Trick*

As described at the beginning of this section, a key bottleneck of standard Bayesian approaches is the potentially high dimensionality of the dynamics parameter vector $\boldsymbol{\theta}$. The same is true for our approach. If we were to keep track of the distributions over all parameters of our dynamics model, calculating $p_D(\dot{\mathcal{X}}|\mathcal{D}, \mathcal{T})$ quickly becomes infeasible.

However, especially in the case of modeling $\boldsymbol{f}$ with a neural network, the benefits of keeping distributions directly over $\boldsymbol{\theta}$ is unclear due to over-parametrization. For both the downstream tasks and our training method, we are mainly interested in the distributions in the state space. Usually, the state space is significantly lower dimensional compared to the parameter space of $\boldsymbol{\theta}$. Furthermore, since the exact posterior state distributions are generally intractable, they normally have to be approximated anyways with simpler distributions for downstream tasks [Sch+15; Hou+16; Ber+17]. Thus, we change the parametrization of our dynamics model as follows. Instead of working directly with $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\theta})$ and keeping a distribution over $\boldsymbol{\theta}$, we model uncertainty directly on the level of the vector field as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\psi}) + \boldsymbol{\Sigma}_D^{\frac{1}{2}}(\boldsymbol{x}(t), \boldsymbol{\psi})\boldsymbol{\epsilon}, \tag{4.10}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_K)$ is drawn once per rollout (i.e., fixed within a trajectory) and $\boldsymbol{\Sigma}_D$ is a state-dependent and positive semi-definite matrix parametrized by a neural network. Here, $\boldsymbol{\psi}$ are the parameters of the new dynamics model, consisting of both the original parameters $\boldsymbol{\theta}$ and the weights of the neural network parametrizing $\boldsymbol{\Sigma}_D$. To keep the number of parameters reasonable, we employ a weight sharing scheme, detailed in Appendix A.3.2.

In spirit, this modeling paradigm is very closely related to standard Bayesian training of NODEs. In both cases, the random distributions capture a distribution over a set of deterministic, ordinary differential equations. This should be seen in stark contrast to stochastic differential equations,

where the randomness in the state space, i.e., diffusion, is modeled with a stochastic process. In comparison to Equation (4.10), the latter is a time-varying disturbance added to the vector field. In that sense, our model still captures the *epistemic* uncertainty about our system dynamics, while an SDE model captures the intrinsic process noise, i.e., *aleatoric* uncertainty. While this reparametrization does not allow us to directly calculate $p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$, we obtain a Gaussian distribution for the marginals $p_D(\dot{\boldsymbol{x}}_{\text{supp}}|\boldsymbol{x}_{\text{supp}})$. To retrieve $p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$, we use the smoother model's predictive state posterior to obtain

$$p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T}) = \int p_D(\dot{\boldsymbol{x}}_{\text{supp}},\boldsymbol{x}_{\text{supp}}|\mathcal{D},\mathcal{T})d\boldsymbol{x}_{\text{supp}} \tag{4.11}$$

$$\approx \int p_D(\dot{\boldsymbol{x}}_{\text{supp}}|\boldsymbol{x}_{\text{supp}})p_S(\boldsymbol{x}_{\text{supp}}|\mathcal{T},\mathcal{D})d\boldsymbol{x}_{\text{supp}}. \tag{4.12}$$

4.2.4 *Comparing Gradient Distributions via the Wasserstein Distance*

To compare and eventually match $p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$ and $p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$, we propose to use the Wasserstein distance [Kan39], since it allows for an analytic, closed-form representation, and since it outperforms similar measures (like forward, backward and symmetric KL divergence) in our exploratory experiments. The squared type-2 Wasserstein distance gives rise to the term

$$\mathbb{W}_2^2\left[p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T}),p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})\right] = \mathbb{W}_2^2\left[p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T}),\mathbb{E}_{\boldsymbol{x}_{\text{supp}}\sim p_{\text{GP}}(\boldsymbol{x}_{\text{supp}}|\mathcal{D},\mathcal{T})}\left[p_D(\dot{\boldsymbol{x}}_{\text{supp}}|\boldsymbol{x}_{\text{supp}})\right]\right], \tag{4.13}$$

that we will later use to regularize the smoothing process. To render the calculation of this regularization term computationally feasible, we introduce two approximations. First, observe that an exact calculation of the expectation in Equation (4.13) requires mapping a multivariate Gaussian through the deterministic neural networks parametrizing $\boldsymbol{f}$ and $\boldsymbol{\Sigma}_D$ in Equation (4.10). To avoid complex sampling schemes, we carry out a certainty-equivalence approximation of the expectation, that is, we evaluate the dynamics model on the posterior smoother mean $\boldsymbol{\mu}_{\text{S, supp}}$. As a result of this approximation, observe that both $p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$ and $p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})$ become Gaussians. However, the covariance structure of these matrices is very different. Since we use independent GPs for different state dimensions, the smoother only models the covariance between the state values within the same dimension, across different time points. Furthermore, since $\boldsymbol{\epsilon}$, the random variable that captures the randomness of the dynamics across all

time-points, is only K-dimensional, the covariance of p_D will be degenerate. Thus, we do not match the distributions directly, but instead match the marginals of each state coordinate at each time point independently at the different supporting time points. Hence, using first marginalization and then the certainty equivalence, Equation (4.13) reduces to

$$\begin{aligned}
&\mathbb{W}_2^2\left[p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T}), p_D(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T})\right] \qquad (4.14)\\
&\quad\approx \sum_{k=1}^{K}\sum_{i=1}^{|\dot{\mathcal{X}}|} \mathbb{W}_2^2\left[p_S(\dot{x}^{(k)}_{\text{supp}}(t_{\text{supp},i})|\mathcal{D},\mathcal{T}), p_D(\dot{x}^{(k)}_{\text{supp}}(t_{\text{supp},i})|\mathcal{D},\mathcal{T})\right]\\
&\quad\approx \sum_{k=1}^{K}\sum_{i=1}^{|\dot{\mathcal{X}}|} \mathbb{W}_2^2\left[p_S(\dot{x}^{(k)}_{\text{supp}}(t_{\text{supp},i})|\mathcal{D},\mathcal{T}), p_D(\dot{x}^{(k)}_{\text{supp}}(t_{\text{supp},i})|\boldsymbol{\mu}_{\text{S, supp}})\right].
\end{aligned}$$

Conveniently, the Wasserstein distance can now be calculated analytically, since for two one-dimensional Gaussians

$$a \sim \mathcal{N}(\mu_a, \sigma_a^2) \qquad (4.15)$$

and

$$b \sim \mathcal{N}(\mu_b, \sigma_b^2), \qquad (4.16)$$

we have

$$\mathbb{W}_2^2[a,b] = (\mu_a - \mu_b)^2 + (\sigma_a - \sigma_b)^2. \qquad (4.17)$$

4.2.5 *Final Loss Function*

As explained in the previous paragraphs, distributional gradient matching trains a smoother regularized by a dynamics model. Both the parameters of the smoother $\boldsymbol{\varphi}$, consisting of the trainable parameters of the GP prior mean $\boldsymbol{\mu}$, the feature map ϕ, and the kernel $\mathcal{K}$, and the parameters of the dynamics model ψ are trained concurrently, using the same loss function. This loss consists of two terms, of which the regularization term was already described in Equation (4.14). While this term ensures that the smoother follows the dynamics, we need a second term ensuring that the smoother also follows the data. To this end, we follow standard GP regression literature, where it is common to learn the GP hyperparameters

by maximizing the marginal log likelihood of the observations, i.e. $\mathcal{L}_{\text{data}}$ [Ras04b]. Combining these terms, we obtain the final objective

$$\mathcal{L}(\varphi, \psi) \coloneqq \mathcal{L}_{\text{data}} - \lambda \cdot \sum_{k=1}^{K} \sum_{i=1}^{|\dot{\mathcal{X}}|} \mathbb{W}_2^2 \left[p_S(\dot{x}_{\text{supp}}^{(k)}(t_{\text{supp},i}) | \mathcal{D}, \mathcal{T}), p_D(\dot{x}_{\text{supp}}^{(k)}(t_{\text{supp},i}) | \boldsymbol{\mu}_{S,\,\text{supp}}) \right].$$

This loss function is a multi-criteria objective, where fitting the data (via the smoother) and identifying the dynamics model (by matching the marginals) regularize each other.

In our preliminary experiments, we found the objective to be quite robust w.r.t. different choices of λ. In the interest of simplicity, we thus set it in all our experiments in Section 4.3 to a default value of $\lambda = \frac{|\mathcal{D}|}{|\dot{\mathcal{X}}|}$, accounting only for the possibility of having different numbers of supporting points and observations. One special case worth mentioning is $\lambda \to 0$, which corresponds to conventional sequential smoothing, where the second part would be used for identification in a second step, as proposed by Pillonetto and De Nicolao [PD10]. However, as can be seen in Figure 4.4, the smoother fails to properly identify the system without any knowledge about the dynamics and thus fails to provide meaningful state or derivative estimates. Thus, especially in the case of sparse observations, joint training is strictly superior.

In its final form, unlike its pure Bayesian counterparts, DGM does not require any prior knowledge about the system dynamics. Nevertheless, if some prior knowledge is available, one could add an additional, additive term $\log(p(\psi))$ to $\mathcal{L}(\varphi, \psi)$. It should be noted however that this was not done in any of our experiments, and excellent performance can be achieved without.

4.3 EXPERIMENTS

We now compare DGM against state-of-the-art methods. In a first experiment, we demonstrate the effects of an overparametrized, simple dynamics model on the performance of DGM as well as traditional, MC-based algorithms SGLD (Stochastic Gradient Lengevin Dynamics, [WT11]) and SGHMC (Stochastic Gradient Hamiltonian Monte Carlo, [CFG14]). We select our baselines based on the results of Dandekar et al. [Dan+21], who demonstrate that both a variational approach and NUTS (No U-Turn Sampler, Hoffman and Gelman [HG14]) are inferior to these two. Subsequently,

we will investigate and benchmark the ability of DGM to correctly identify neural dynamics models and to generalize across different initial conditions. Since SGLD and SGHMC reach their computational limits in the generalization experiments, we compare against Neural ODE Processes (NDP). Lastly, we will conclude by demonstrating the necessity of all of its components. For all comparisons, we use the julia implementations of SGLD and SGHMC provided by Dandekar et al. [Dan+21], the pytorch implementation of NDP provided by Norcliffe et al. [Nor+21], and our own JAX [Bra+18] implementation of DGM.

4.3.1 *Setup*

As benchmarks, we again consider the two-dimensional *Lotka Volterra (LV)* system (as introduced in Section 2.7, the three-dimensional, chaotic *Lorenz (LO)* system, a four-dimensional *double pendulum (DP)* and a 12-dimensional *quadrocopter (QU)* model. For all systems, the exact equations and ground truth parameters are restated in the appendix, Section A.3.1. For each system, we create two different data sets. In the first, we include just one densely observed trajectory, taking the computational limitations of the benchmarks into consideration. In the second, we include many, but sparsely observed trajectories (5 for LV and DP, 10 for LO, 15 for QU). This setting aims to study generalization over different initial conditions.

4.3.2 *Metric*

We use the log likelihood as a metric to compare the accuracy of our probabilistic models. In the 1-trajectory setting, we take a grid of 100 time points equidistantly on the training trajectory. We then calculate the ground truth and evaluate its likelihood using the predictive distributions of our models. When testing for generalization, we repeat the same procedure for unseen initial conditions.

4.3.3 *Effects of Overparametrization*

We first study a three-dimensional, linear system of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t), \tag{4.18}$$

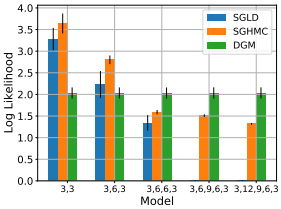


FIGURE 4.2: SGLD does not converge for strongly overparametrized models, the performance of SGHMC deteriorates. DGM is not noticeably affected.

where $\boldsymbol{A}$ is a randomly chosen matrix with one stable and two marginally stable eigenvalues. For the dynamics model, we choose a linear Ansatz

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{B}\boldsymbol{x}(t), \tag{4.19}$$

where $\boldsymbol{B}$ is parametrized as the product of multiple matrices. The dimension of the matrices of each factorization are captured in a string of integers of the form $(3, a_1, \ldots, a_J, 3)$. For example, $(3, 3)$ corresponds to $\boldsymbol{B}$ being just one matrix, while $(3, 6, 6, 3)$ corresponds to $\boldsymbol{B} = \boldsymbol{B}_1\boldsymbol{B}_2\boldsymbol{B}_3$, with $\boldsymbol{B}_1 \in \mathbb{R}^{3\times 6}$, $\boldsymbol{B}_2 \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{B}_3 \in \mathbb{R}^{6\times 3}$. All of these models can be interpreted as linear neural networks, forming a simple case of the nonparametric systems we study later. Unlike general neural networks, the expressiveness of the Ansatz is independent of the number of parameters, allowing us to isolate the effects of overparametrization. In Figure 4.2, we show the mean and standard deviation of the log likelihood of the ground truth over 10 different noise realizations. The exact procedure for one noise realization is described in the appendix, Section A.3.2. While SGLD runs into numerical issues after a medium model complexity, the performance of SGHMC continuously disintegrates, while DGM is unaffected. This foreshadows the results of the next two experiments, where we observe that the MC-based approaches are not suitable for the more complicated settings.

4.3.4 *Single Trajectory Benchmarks*

In Table 4.1, we evaluate the log-likelihood of the ground truth for the four benchmark systems, obtained when learning these systems using a neural ODE as a dynamics model (for more details, see Section A.3.2.1). Clearly, DGM performs the best on all systems, even though we supplied both SGLD and SGHMC with very strong priors and fine-tuned them with an extensive hyperparameter sweep (see Section A.3.3. Despite this effort, we

	Log Likelihood		
	DGM	SGHMC	SGLD
LV 1	$\mathbf{1.96 \pm 0.21}$	1.36 ± 0.0693	1.03 ± 0.0581
LO 1	$\mathbf{-0.57 \pm 0.11}$	-3.02 ± 0.158	-2.67 ± 0.367
DP 1	$\mathbf{2.13 \pm 0.14}$	1.88 ± 0.0506	1.85 ± 0.0501
QU 1	$\mathbf{0.64 \pm 0.07}$	-5.00 ± 1.36	NaN
	Prediction time [ms]		
	DGM	SGHMC	SGLD
LV 1	$\mathbf{0.68 \pm 0.04}$	14.98 ± 0.23	14.59 ± 0.15
LO 1	$\mathbf{0.99 \pm 0.05}$	$98.93. \pm 5.79$	105.03 ± 12.22
DP 1	$\mathbf{1.31 \pm 0.05}$	10.60 ± 0.21	11.34 ± 0.76
QU 1	$\mathbf{3.76 \pm 0.12}$	24.68 ± 6.58	NaN

TABLE 4.1: Log likelihood and prediction times of 100 ground truth sample points, with mean and standard deviation taken over 10 independent noise realizations, for neural ODEs trained on a single, densely sampled trajectory.

failed to get SGLD to work on Quadrocopter 1, where it always returned NaNs. This is in stark contrast to DGM, which performs reliably without any pre-training or priors.

4.3.5 *Prediction speed*

To evaluate prediction speed, we consider the task of predicting 100 points on a previously unseen trajectory. To obtain a fair comparison, all algorithms' prediction routines were implemented in JAX [Bra+18]. Furthermore, while we used 1000 MC samples when evaluating the predictive posterior for the log likelihood to guarantee maximal accuracy, we only used 200 samples in Table 4.1. Here, 200 was chosen as a minimal sample size guaranteeing reasonable accuracy, following a preliminary experiment visualized in Appendix A.3.3. Nevertheless, the predictions of DGM are 1-2 orders of magnitudes faster, as can be seen in Table 4.1. This further illustrates the advantage of relying on a smoother instead of costly, numerical integration to obtain predictive posteriors in the state space.

	Log Likelihood	
	DGM	NDP
LV 100	$\mathbf{1.81 \pm 0.08}$	0.62 ± 0.27
LO 125	$\mathbf{-2.18 \pm 0.76}$	-2.85 ± 0.05
DP 100	$\mathbf{1.86 \pm 0.05}$	0.88 ± 0.05
QU 64	$\mathbf{-0.54 \pm 0.36}$	-0.91 ± 0.07

TABLE 4.2: Log likelihood of 100 ground truth sample points, with mean and covariance taken over 10 independent noise realizations, for neural ODEs trained on a multiple, sparsely sampled trajectory.The number following the system name denotes the number of trajectories in the training set.

4.3.6 *Multi-Trajectory Benchmarks*

Next, we take a set of trajectories starting on an equidistant grid of the initial conditions. Each trajectory is then observed at 5 equidistant observation times for LV and DP, and 10 equidistant observation times for the chaotic Lorenz and more complicated Quadrocopter. We test generalization by randomly sampling a new initial condition and evaluating the negative log likelihood of the ground truth at 100 equidistant time points. In Table 4.2, we compare the generalization performance of DGM against NDP, since despite serious tuning efforts, the MC methods failed to produce meaningful results in this setting. DGM clearly outperforms NDP, a fact which is further exemplified in Figure 4.3. There, we show the test log likeliood for Lotka Volterra trained on an increasing set of trajectories. Even though the time grid is fixed and we only decrease the distance between initial condition samples, the dynamics model helps the smoother to generalize across time as well. In stark contrast, NDP fails to improve with increasing data after an initial jump.

4.3.7 *Ablation study*

We next study the importance of different elements of our approach via an ablation study on the Lorenz 125 dataset, shown in Figure 4.4. Comparing the two rows, we see that joint smoothing across trajectories is essential to transfer knowledge between different training trajectories. Similarly,

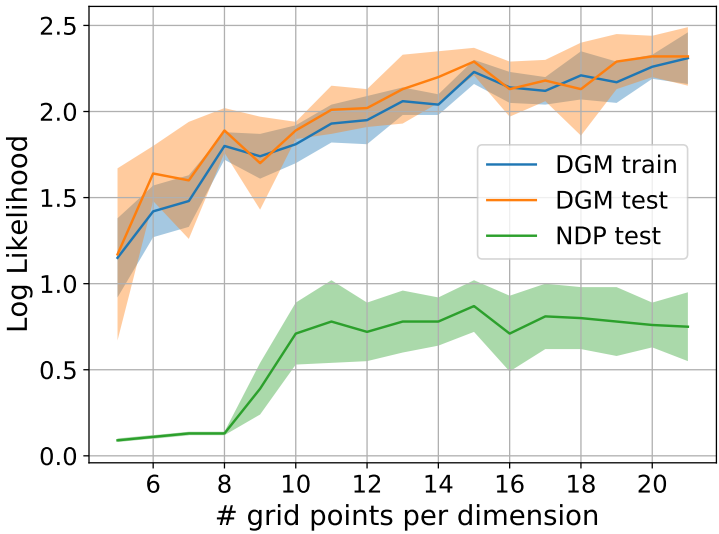


FIGURE 4.3: Log likelihood of the ground truth for Lotka Volterra for increasing number of trajectories with 5 observations each.

comparing the two columns, we see that the dynamics model enables the smoother to reduce its uncertainty in between observation points.

4.3.8 *Computational Requirements*

For the one trajectory setting, all DGM related experiments were run on a Nvidia RTX 2080 Ti, where the longest ones took 15 minutes. The comparison methods were given 24h, on Intel Xeon Gold 6140 CPUs. For the multi-trajectory setting, we used Nvidia Titan RTX, where all experiments finished in less than 3 hours. A more detailed run time compilation can be found in Section A.3.4. Using careful implementation, the run time of DGM scales linearly in the number of dimensions K. However, since we use an accurate RBF kernel for all our experiments reported in this section, we have cubic run time complexity in $\sum_{m=1}^{M} N_m$. In principle, this can be alleviated by deploying standard feature approximation methods [RR+07; Liu+20], or by combining it with SLEIPNIR, as introduced in the previous section. While this is a well known fact, we will talk about it in more detail

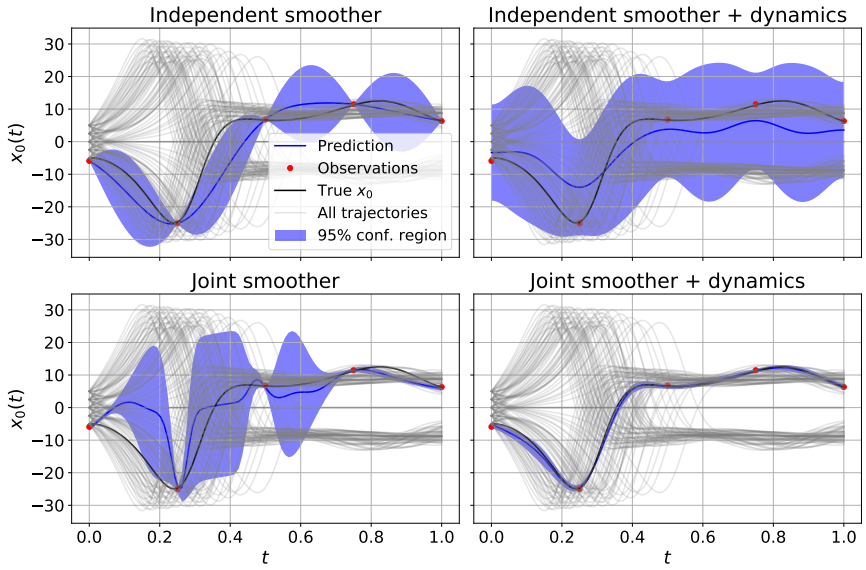


FIGURE 4.4: Illustration of DGM: Learning a joint smoother (first vs second row) across trajectories enables sharing observational data. Dynamics regularization (first vs second column) substantially improves prediction accuracy of joint smoother.

in the next section, where we demonstrate how to apply DGM to a realistic, real-world data set.

4.3.9 *Scaling to many observations or trajectories*

Let N be the total number of observations, summed over all training trajectories. In this section, we will analyze the computational complexity of DGM in terms of N and demonstrate how this can be drastically reduced using standard methods from the literature. This will be a crucial first step towards enabling the case study in Section 4.4.

For notational compactness, we will assume that the supporting points in $\mathcal{T}$ are at the same locations as the observations in $\mathcal{D}$. However, this is by no means necessary. As long as they are chosen to be constant or stand in a linear relationship to the number of observations, our analysis still holds. We will thus use $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{supp}}$ and the corresponding quantities

interchangeably. Similarly, we will omit the k that was used for indexing the state dimension and assume one-dimensional systems. The extension to multi-dimensional systems is straight forward and comes at the cost of an additional factor K.

Fortunately, most components of the loss of DGM given by Equation (4.3) can be calculated in linear time. In particular, it is worth noting that the independence assumption made when calculating the Wasserstein distance in Equation (4.14) alleviates the need to work with the full covariance matrix and lets us work with its diagonal elements instead. Nevertheless, there are several terms that are not straight forward to calculate. Besides the marginal log likelihood of the observations, these are the posteriors

$$p_S(\boldsymbol{x}|\mathcal{D},\mathcal{T}) = \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_{\text{post}},\boldsymbol{\Sigma}_{\text{post}}\right), \tag{4.20}$$

$$p_S(\dot{\mathcal{X}}|\mathcal{D},\mathcal{T}) = \mathcal{N}\left(\dot{\boldsymbol{x}}_{\text{supp}}|\boldsymbol{\mu}_S,\boldsymbol{\Sigma}_S\right), \tag{4.21}$$

where

$$\boldsymbol{\mu}_{\text{post}} = \boldsymbol{\mu} + \boldsymbol{\mathcal{K}}^T(\boldsymbol{\mathcal{K}} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu}), \tag{4.22}$$

$$\boldsymbol{\Sigma}_{\text{post}} = \boldsymbol{\mathcal{K}} - \boldsymbol{\mathcal{K}}^T(\boldsymbol{\mathcal{K}} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{\mathcal{K}}, \tag{4.23}$$

$$\boldsymbol{\mu}_S = \dot{\boldsymbol{\mu}} + \dot{\boldsymbol{\mathcal{K}}}(\boldsymbol{\mathcal{K}} + \sigma^2\boldsymbol{I})^{-1}\left(\boldsymbol{y} - \boldsymbol{\mu}\right), \tag{4.24}$$

$$\boldsymbol{\Sigma}_S = \ddot{\boldsymbol{\mathcal{K}}} - \dot{\boldsymbol{\mathcal{K}}}(\boldsymbol{\mathcal{K}} + \sigma^2\boldsymbol{I})^{-1}\dot{\boldsymbol{\mathcal{K}}}^\top. \tag{4.25}$$

Here, Equation (4.20) is used for prediction, while its mean is also used in the approximation of Equation (4.14). On the other hand, Equation (4.21) is used directly for Equation (4.14). Note that in both cases, we only need the diagonal elements of the covariance matrices, a fact that will become important later on.

In its original form, calculating the matrix inverses of both Equation (4.20) and Equation (4.21) has cubic complexity in N. To alleviate this problem, we follow Rahimi, Recht, et al. [RR+07] and Angelis et al. [Ang+20] by using a feature approximation of the kernel matrix and its derivatives. In particular, let $\boldsymbol{\Phi} \in \mathbb{R}^{F\times N}$ be a matrix of F random Fourier features as described by Rahimi, Recht, et al. [RR+07]. Furthermore, denote $\dot{\boldsymbol{\Phi}}$ as its derivative w.r.t. the time input variable, as defined by Angelis et al. [Ang+20]. We can now approximate the kernel matrix and its derivative versions as

$$\boldsymbol{K} \approx \boldsymbol{\Phi}^\top\boldsymbol{\Phi}, \quad \dot{\boldsymbol{\mathcal{K}}}^\top \approx \dot{\boldsymbol{\Phi}}^\top\boldsymbol{\Phi}, \quad \text{and} \quad \ddot{\boldsymbol{\mathcal{K}}} \approx \dot{\boldsymbol{\Phi}}^\top\dot{\boldsymbol{\Phi}}. \tag{4.26}$$

Using these approximations, we can leverage the Woodbury idendity to approximate

$$(\boldsymbol{\mathcal{K}} + \sigma^2 \boldsymbol{I})^{-1} \approx \frac{1}{\sigma^2}\left[\boldsymbol{I} - \boldsymbol{\Phi}^\top \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}\right]. \tag{4.27}$$

This approximation allows us to invert a $F \times F$ matrix, to replace the inversion of a $N \times N$ matrix. This can be leveraged to calculate

$$\boldsymbol{\mu}_S = \dot{\boldsymbol{\mu}} + \dot{\boldsymbol{\mathcal{K}}}(\boldsymbol{\mathcal{K}} + \sigma^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) \tag{4.28}$$

$$\approx \dot{\boldsymbol{\mu}} + \frac{1}{\sigma^2}\dot{\boldsymbol{\Phi}}^\top \boldsymbol{\Phi}\left[\boldsymbol{I} - \boldsymbol{\Phi}^\top \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}\right](\boldsymbol{y} - \boldsymbol{\mu}) \tag{4.29}$$

and

$$\boldsymbol{\Sigma}_S = \ddot{\boldsymbol{\mathcal{K}}} - \dot{\boldsymbol{\mathcal{K}}}(\boldsymbol{\mathcal{K}} + \sigma^2 \boldsymbol{I})^{-1}\dot{\boldsymbol{\mathcal{K}}}^\top \tag{4.30}$$

$$\approx \dot{\boldsymbol{\Phi}}^\top \dot{\boldsymbol{\Phi}} - \frac{1}{\sigma^2}\dot{\boldsymbol{\Phi}}^\top \boldsymbol{\Phi}\left[\boldsymbol{I} - \boldsymbol{\Phi}^\top \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}\right]\boldsymbol{\Phi}^\top \dot{\boldsymbol{\Phi}}. \tag{4.31}$$

Evaluating the matrix multiplications of Equation (4.29) in the right order leads to a computational complexity of $\mathcal{O}(NF^2 + F^3)$. Similarly, the diagonal elements of the covariance given by Equation (4.21) can be calculated with the same complexity, by carefully summarizing everything in between $\dot{\boldsymbol{\Phi}}^\top$ and $\dot{\boldsymbol{\Phi}}$ as one $F \times F$ matrix and then calculating the N products independently.

Since the components of Equation (4.20) have the exact same form as the components of Equation (4.21), they can be approximated in the exact same way to obtain the exact same computational complexity. Thus, the only components that need further analysis are the components of the marginal log likelihood of the observations, particularly

$$\boldsymbol{y}^\top (\boldsymbol{\mathcal{K}} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y} \approx \boldsymbol{y}^\top \frac{1}{\sigma^2}\left[\boldsymbol{I} - \boldsymbol{\Phi}^\top \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}\right]\boldsymbol{y} \tag{4.32}$$

and

$$\operatorname{logdet}(\boldsymbol{\mathcal{K}} + \sigma^2 \boldsymbol{I}) \approx \operatorname{logdet}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma^2 \boldsymbol{I}) \tag{4.33}$$

$$\approx \operatorname{logdet}(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I}) + (N - F)\log(\sigma^2). \tag{4.34}$$

In the last line, we used the fact that the nonzero eigenvalues of the transposed of a matrix stay the same.

Combining all these tricks, it is clear that the overall complexity of DGM can be reduced to $\mathcal{O}(NF^2 + F^3)$. Since F is a constant controlling the quality of the approximation scheme and is usually chosen to be constant, we thus get essentially linear computational complexity in the number of observations. Note that these derivations are completely independent of what scheme is chosen to obtain the feature matrix $\mathbf{\Phi}$. For ease of implementation, we opted for random Fourier features though in our experiments.

EXPERIMENTAL PROOF OF CONCEPT To demonstrate that this approximation scheme can be used in the context of DGM, we tested it on the multi-trajectory experiment of Lotka Volterra. To this end, we increased the grid from 10 points per dimension to 25, leading to a total number of 3125 observations instead of 500. As an approximation, we used 50 random Fourier features. Through this approximation, DGM became slightly more sensitive to the optimization hyperparameters. Nevertheless, it reached comparable accuracy within roughly 440 seconds of training, compared to the 408 seconds needed to train the approximation free version on LV 100.

4.4 CASE STUDY: USHCN

Thus far, this thesis investigated gradient matching techniques in the context of theoretical, simulated data sets. This allowed us to guarantee key assumptions on the underlying dynamical system, like Markovianity or time-independence. In the following section, we want to investigate how DGM performs on a real world data set, where we cannot know if these assumptions hold or if they might be violated. The code for this section is publicly available on github[1].

4.4.1 *The data set*

For this practical case study, we use the publicly available daily data set of the United State Historical Climatology Network [USHCN, MWV15]. This data set contains five measurements: Maximum daily temperature, minimum daily temperature, percipitation, snowfall, snowdepth. It contains measurements over 150 years for 1,218 meteorological stations scattered over the United States. To clean the data, we follow the procedure described by De Brouwer et al. [De +19], by using their code published alongside the

1 https://github.com/lasgroup/ushcn_dgm

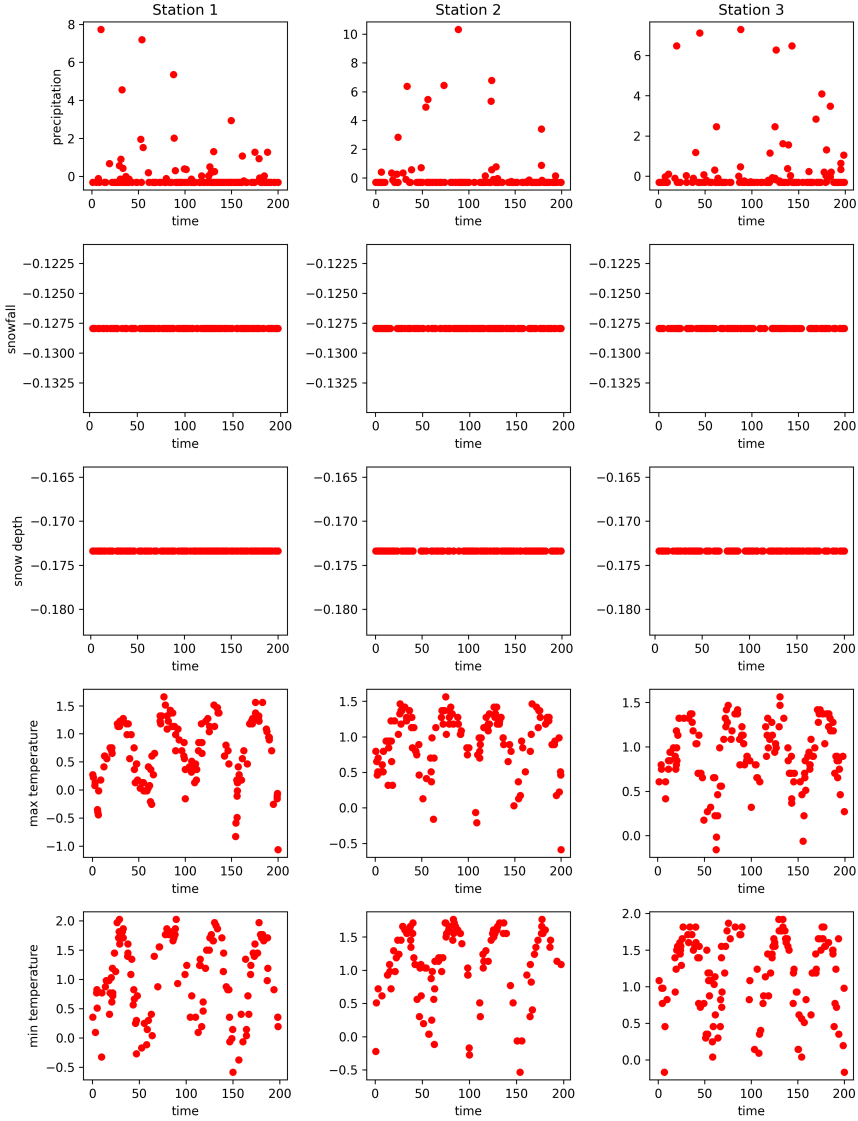


FIGURE 4.5: Visualization of three trajectories of the USHCN dataset after preprocessing.

paper on github. Amongst filtering out trajectories with insufficient number of observations, normalizing the measurements, and subsampling the data set, they also restrict the analysis to the four years between 1996 and 2000. A few example trajectories are shown in Figure 4.5.

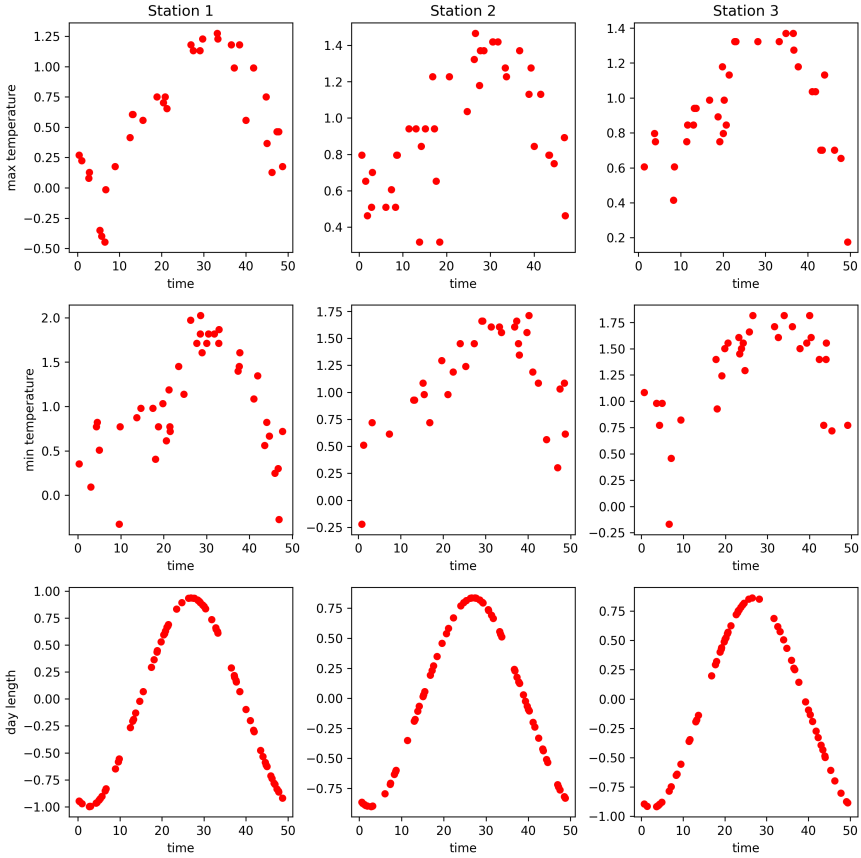


FIGURE 4.6: Visualization of three trajectories of the USHCN dataset after removing snow and percipitation and including daylength.

4.4.2 *Preparing the data set*

To deploy DGM, we explicitly assume that there exists an underlying, dynamical system that is responsible for creating the observations shown

in Figure 4.5. We then extract information about this system from the data to be used to extrapolate either over time or over locations.

Intuitively, it might be reasonable to assume the existing of underlying meteorological dynamics that drive the observed variables. However, when looking at Figure 4.5, we observe a few key issues.

On most days, it is not raining. When it rains, it rains a lot. Since the data has been subsampled, there is rarely a direct observation before or after a rainy day. Thus, a rainy day comes as a surprise and appears to be more an outlier than a feature. In terms of information theory, we postulate that the rain dynamics are too fast for the sampling frequency chosen and thus cannot be distinguished from being outliers. Thus, precipitation is not included in our case study.

A similar thing can be observed for snowfall and snow depth. For most stations, snowfall and snow depth stay constant (on a value that corresponds to 0, in the not normalized data). Similarly, where snowfall exists, it follows the same seemingly stochastic behavior as rainfall. Again, since neither predicting outliers nor constants is of particular interest for our case, we omit these two variables.

Some preliminary experiments with the GRU algorithm by De Brouwer et al. [De +19], on whom we already relied for the data preprocessing, confirm this notion. Even though we train GRU with precipitation and snowfall included, the final predictions are not significantly better than just predicting constants.

After removing snowfall, precipitation and snow depth, we are left with predicting the minimum and maximum daily temperature. Here, the sampling frequency seems sufficient to allow for an interesting case study. However, we see a strong correlation between the two states. Indeed, it seems reasonable to assume that with the chosen sampling frequency, min and max temperature are mainly dominated by seasonal trends. And such trends should affect both the minimum and maximum daily temperature in a similar way, except for noise.

Unfortunately, this leaves us with a data set that does not seem to be Markovian, since there seem to exist multiple points where the states are the same, but the derivatives are very different. Trying to fit such a system with a normal ODE would just lead to the prediction of a constant. One way of handling such a system is to make the dynamics time-dependent. However, while this could work well in an interpolation, we would lose a key advantage of DGM. If the dynamics were time-dependent, it is no longer reasonable to assume that the resulting system could extrapolate

into the future, since it has no information about the dynamics at these time points.

Thus, we chose to go a different route. In the meta-data of the data set, we get 3D coordinates of every weather station. By using the astral package[2], such locations can easily be turned into the length of day at this location. For each station, the day length is calculated for all time points where we have either a min or a max temperature measurement. The resulting data is visualized in Figure 4.6. While day length and temperature seem to have a similar shape, there seems to be a phase difference. Thus, the system now lives on a 2D manifold. Thus, many if not all of the previously pathological points will now be separated via the newly introduced state. Since location is always available and easy to obtain in any real world setting, day length will be included in all of our experiments in this section.

Lastly, it should be noted that including the location of the weather station has another, very important benefit. In previous experiments, we we let the smoother depend on the noise free initial condition. This was key in demonstrating the capability of DGM to generalize over initial conditions of a dynamical system.

For USHCN however, no noise free initial conditions are available. Thus, both training and prediction become much more difficult. However, we can alleviate this problem by observing one key fact: In the framework of DGM, there is no mechanism that enforces that the smoother input needs to be the actual initial condition in a dynamical system sense. Instead, the smoother input just needs to be something that uniquely identifies different dynamical systems. Of course, it should carry some information about the behavior of the system, over which the smoother can generalize. Both of these criteria are clearly met by the station's locations. A location is clearly a unique identifier for a station. And it is not unreasonable to assume that weather phenomena are somehow similar for similar locations. Thus, instead of trying to obtain initial conditions, we directly feed the locations to the smoother. This side-steps various challenges associated with trying to denoise the initial conditions. And it preserves all the capabilities of DGM, including generalization over both time and locations.

4.4.3 *Challenge of the data set*

In reality, assuming the existence of an underlying ODE might be wrong for some systems. In such a case, the additional regularization via the

2 https://astral.readthedocs.io/en/latest/index.html

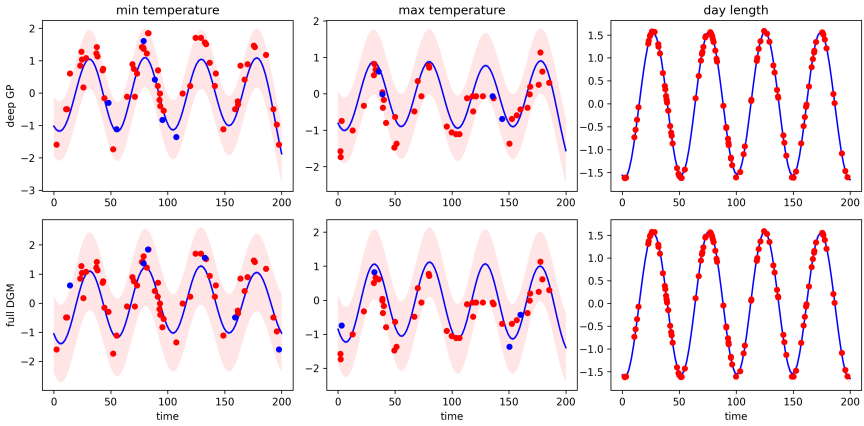


FIGURE 4.7: Predictions on a trajectory with training observations (red dots), where some observations were hold out for testing (blue dots). The mean of the smoother is shown in a blue line, with 95% confidence intervals of the pure smoother model (blue shaded, barely visible) and a confidence interval combining smoother uncertainty and observation noise levels (red shaded).

	time	initial condition
deep GP	**0.649 +- 0.0179**	**0.626 +- 0.0499**
full DGM	0.742 +- 0.0180	0.752 +- 0.0439

TABLE 4.3: Mean and standard deviations of the negative test log likelihood for the dynamics free smoother model (deep GP) and the full, dynamics-regularized DGM model. When calculating the nlls, the contribution of the third state was ignored, as it is an artificially calculated, noise-free state.

dynamics model provided by DGM might actually hurt the performance of the smoother. To increase the flexibility of the model without sacrificing generalization capability, we provide the dynamics model with the locations input as well. However, if the location is treated as a constant state, the dynamics model is still time-independent and Markovian.

In this section, we aim to get a first understanding of how these assumptions influence the performance of DGM on the USHCN data. To this end, we deploy DGM in two settings similar to the experiments we performed

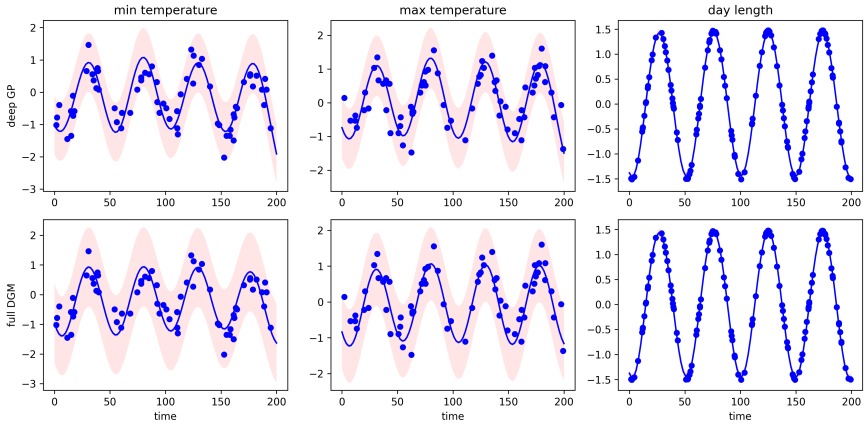


FIGURE 4.8: Predictions on a completely new trajectory. Since no training observations are available, all observations are used for testing (blue dots). The mean of the smoother is shown in a blue line, with 95% confidence intervals of the pure smoother model (blue shaded, barely visible) and a confidence interval combining smoother uncertainty and observation noise levels (red shaded).

in Section 4.3 on simulated data. On a subset of 200 trajectories from the USHCN data set, we test how DGM generalizes to unseen times and unseen initial conditions.

To test time generalization, we randomly subsample the observation times of all 200 trajectories. For each trajectory, 10% of the data points are kept for testing, while the other 90% are used for training. Then, since no ground truth is available, we calculate the test negative log likelihood of the remaining 10% of data points. Here, both the observation noise and the uncertainty of the GP are considered in the Gaussian prediction.

To test generalization across initial conditions, we then sample 10 trajectories out of the remaining trajectories that have not been used for training. Again, since only the noisy observations are available for these trajectories, we calculate the negative log likelihood of the whole, as of now unseen trajectory.

Both settings are executed 10 times for 10 different sets of 200 randomly sampled trajectories. Optimization parameters like learning rate and number of training steps have been chosen in a preliminary step. This optimization was guided purely by the visual representation of the

learning curve, there was no numerical optimization done. As for the hyperparameters "weight decay smoother" and "weight decay dynamics", the test negative log likelihood was optimized via a grid search on the grid $[0, 10e-6, 10e-5, 10e-4]$. For this purpose, a new set of 200 trajectories was used, that are not used in the sweeps later on.

For each of the 10 trajectory sets, multiple training steps were performed sequentially. First, the smoother is trained via $\mathcal{K}_{\text{data}}$, while keeping the parameters of the dynamics model constant (smoother pre-training). Then, the dynamics model is trained using the full DGM loss, but while keeping the smoother parameters constant. Finally, in the last step, smoother and dynamics parameters are trained jointly, by using the full DGM loss.

This pre-training stabilizes the training procedure. However, more importantly, it allows us to directly compare the performance of the DGM model to a normal GP smoother trained just with $\mathcal{L}_{\text{data}}$. In Figure 4.7, we show one example trajectory for the time interpolation experiment. Here, one of the training trajectories is shown with the corresponding points that have been left out for training. In Figure 4.8, we then show the predictions of the same model on a previously unknown initial condition. In both cases, the performance of the dynamics-free smoother and the dynamics-informed DGM do not seem to be much different visually. The difference becomes clear when looking at the negative test log likelihoods in Table 4.3. There, we present the mean and the standard deviation of the negative test log likelihoods taken over all ten different data set splits. It becomes clear that the smoother outperforms the dynamics-informed approach in both cases. And in both cases, the difference is statistically significant.

This statistically significant difference in performance seems to suggest that the additional regularization provided by the dynamics model is hurting the performance of the smoother model. Thus, we can conclude that there are some effects present in this real world data set, that cannot be captured with our time-independent, Markovian dynamics. The additional flexibility provided by letting the dynamics model depend on the station's locations was not sufficient to fully capture the different dynamics.

4.4.4 *Including dynamics helps with extrapolation*

However, even though not all assumptions seem to be satisfied, there are tasks where the dynamics model can play a crucial role. The key strength of a dynamics model is extrapolation over time. In this section, we will

demonstrate how the dynamics model can be deployed in an extrapolation setting to strongly improve the performance of the smoother.

To this end, we will differentiate between two uses of the dynamics model. In standard DGM, the dynamics model is used to regularize the smoother model during training. However, at prediction, the dynamics model is not deployed anymore and DGM purely focuses on the smoother predictions. As we demonstrate in this section, this can hurt extrapolation performance. Thus, we also demonstrate a second use of the dynamics model, which is deploying it at prediction via numerical integration.

The performance of these DGM variants will be analyzed in the setting of extrapolation over time. In this setting, we train our models on the first year of data for randomly sampled 200 trajectories. Then, we evaluate the negative log likelihood on the second year of data on the same 200 trajectories.

As in the previous section, we first train just the smoother parameters on $\mathcal{L}_{\text{data}}$, while keeping the dynamics parameters constant. Then, we train only the dynamics parameters on the whole DGM loss, while keeping the smoother parameters constant. At this point, we take a snapshot of our smoother and dynamics models. Note that at this point, the dynamics model has not been used to regularize the smoother, it has merely tried to imitate the smoother to the best of its abilities in the second step. Note that these steps are performed before the proper training with the full DGM loss function. Thus, it will be called pre-training and all models from these snapshot will be indexed with a "p" for pre-training.

After pre-training, the dynamics and smoother parameters are jointly trained using the full DGM loss to obtain a final smoother and dynamics model. During this joint training, the dynamics model regularizes the smoother model via the joint loss function, as described previously. All models obtained from this final training step will be indexed with a "f" for final.

For prediction, we deploy two methodologies. The first method includes no integration. It just uses the predictions of the smoother model, as done in standard DGM. To indicate that these predictions are obtained without integration, it will be indexed with "ni", for no integration. The second method includes numerical integration of the dynamics model. First, we use the smoother model to predict the final point of the first year. This posterior is then sampled to obtain an estimate of the starting point for the next year. After sampling the noise of the dynamics model as well, we can then numerically integrate one example trajectory for the next year.

This process is repeated 100 times to obtain 100 trajectory samples, which are then aggregated to obtain a Gaussian distribution at each time point. To indicate that integration is used to obtain these predictions, we use the index "wi", for with integration. In summary, we thus get four different predictors, summarized in Table 4.4.

nip	smoother predictions of smoother model after pre-training
wip	numerically integrated predictions of models after pre-training
nif	smoother predictions of final smoother model
wif	numerically integrated predictions of models after final training

TABLE 4.4: DGM models deployed in time extrapolation experiment.

To determine the weight decay parameters of DGM, we leave out 5% of the training samples and select the weight decay parameters with the best prediction nll on these left out data.

nip	4.53 +- 3.69
nif	3.45 +- 2.90
wip	1.99 +- 0.800
wif	1.42 +- 0.630

TABLE 4.5: Test negative log likelihood of different DGM snapshots and prediction modes for a year of unseen data in the future. Mean and standard deviation over 10 sets of 200 trajectories independently subsampled from the original data. When calculating the nlls, the contribution of the third state was ignored, as it is an artificially calculated, noise-free state.

The results of this experiment are summarized in Table 4.5. Here, the benefits of the dynamics model are visible two-fold. First, it suggests that including the dynamics model in training is beneficial. Here, we can compare the predictions of the pre-trained and the final smoother model, as well as the predictions of the pre-trained and the final integration predictions. In both cases, choosing the model that leveraged the dynamics regularization (i.e. indexed with "f" instead of "p") yields better test log likelihoods. Second, these results also suggest that including the dynamics model at prediction time is beneficial. Here, we can compare the predictions of the smoother models to the predictions of the integration predictors. For both

the pre-trained models as well as the final models, the integration predictors outperform the smoothers, as indicated by the lower nll of the models with index "wi" as opposed to "ni".

An intuitive understanding of these results can be obtained when inspecting one sample trajectory as shown in Figure 4.9. Here, we show one test case for all four different training / prediction strategies. In the first two rows, we see the predictions of the models without integration. It seems that independently of the dynamics regularization during training, the smoother struggles to extrapolate the oscillating behavior. This is of little surprise. The smoother takes time as an input, but was never trained with times in that range. Thus, it cannot be expected to extrapolate. In contrast, the integration-based predictions in the last two rows already show the oscillatory shapes. Clearly, the dynamics models can generalize to various extent to unseen times. Here, the key difference to the smoother is highlighted. In contrast to the smoother, we explicitly refrain from making the dynamics model time-dependent. Thus, it can be expected to generalize to unseen time ranges.

When comparing the last two rows, we observe that including the dynamics regularization during training does not only increase the accuracy of the dynamics estimates, but also greatly reduces the inherent uncertainty of the models. This observation is interesting, since it highlights the benefits of training smoother and dynamics jointly in the presence of non-Markovian effects. During pre-training, we fit a very flexible smoother to the data. The derivatives of this flexible smoother do not need to follow a dynamics model and might violate underlying assumptions of the dynamics model (as indicated in the previous section). When we then train a dynamics model on this potentially conflicting data, the dynamics model converges to a dynamics model that best explains the conflicting data, potentially leading to errors and overestimation of variance. However, when we regularize the smoother during training with the dynamics model, they jointly converge to a model that balances data fit and the assumptions in the dynamics model. This leads to the better performance shown in the last row.

4.4.5 *Comparisons*

Thus far, we have only presented the performance of our algorithms. To get an understanding of the difficulty of the task, we investigate how different comparison methods perform on this benchmark.

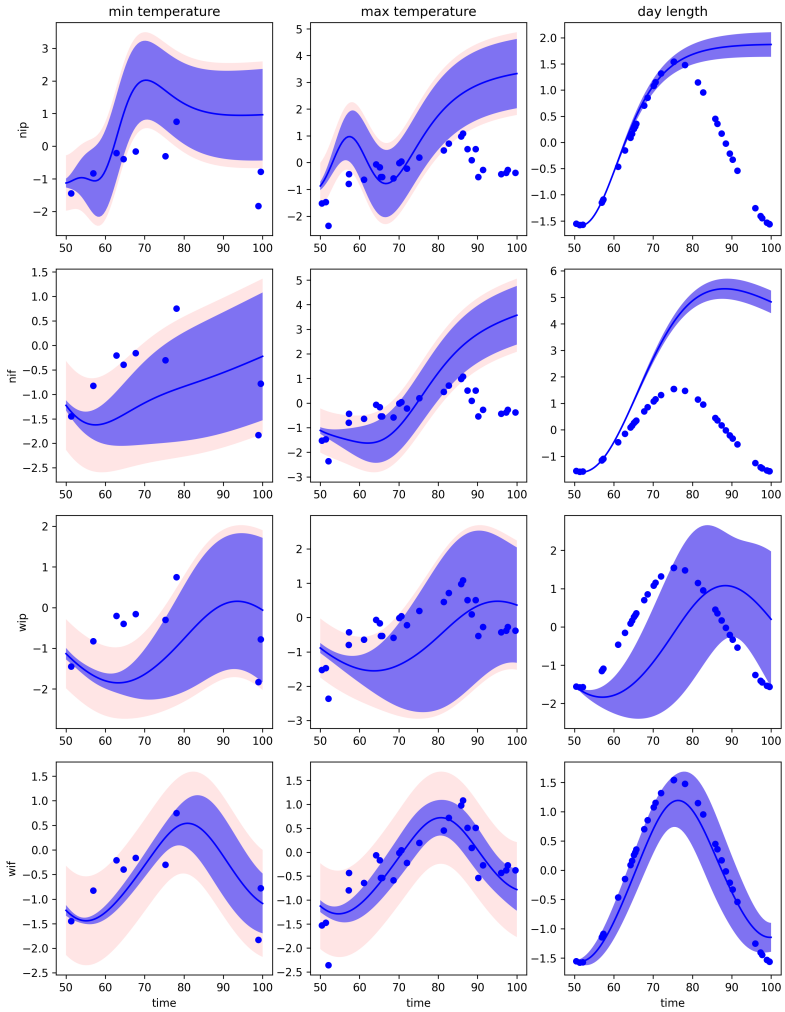


FIGURE 4.9: One sample trajectory of the extrapolation experiment. The test observations are shown as blue dots. The mean and the standard 95% confidence interval of each model is shown in blue, where the shaded red area is combining the uncertainties of the model and the observation noise to a joint 95% confidence interval for the observations.

Since we are still interested in uncertainty aware analysis of dynamics models, we again consider the three state-of-the-art algorithms SGLD,

SGHMC and NDP. Unfortunately, similar to what we already observed in Section 4.3.6, SGLD and SGHMC cannot keep up with the model complexity and the thus resulting numerical challenges. We thus have to restrict our comparisons to NDP.

For NDP, several adaptations were necessary to adapt it to the current setting. In its original form, albeit able to handle non-equidistantly sampled time points, NDP is unable to deal with partially missing observations. However, in the USHCN data, there is almost no time point where all variables are observed concurrently.

Thus, we had to adapt the underlying data structure used in the framework. Previously, NDP worked directly with tuples $(t, \boldsymbol{y})$, e.g. to caluclate the context variables. Clearly, this can not be directly deployed if the observation vector $\boldsymbol{y}$ is missing data. Thus, we changed the data to chunks of the form $(t, y_j, j, \tilde{\boldsymbol{y}}_0)$. Here, j indicates the dimension of the observation that we have currently available and y_j is the j-th entry of the observation vector $\boldsymbol{y}$. This allows us to work directly with missing data. Furthermore, we added the pseudo initial conditions $\tilde{\boldsymbol{y}}_0$ to the vector. These are the same location parameters that we fed to both our smoother and dynamics model in DGM as input, consisting of the 3D GPS coordinates of each station. For practical implementation, we again built on top of their publicly available pytorch implementation. To make it work with the new data structure, we had to make some adaptations. Primarily, this concerned padding and masking operations at different places in the code. These were necessary so that we were still able to use a vectorized representation for efficiency.

Since NDP relies on internal cross-validation, we split the 200 training trajectories into 190 training and 10 validation trajectories at random. This mirrors the 5% testing data we used to optimize the DGM weight-decay parameters.

The resulting performance of NDP is summarized in Table 4.6. Here, a few things should be noted. First, as we can see in the first two rows, NDP trains well and its model translates nicely to the validation trajectories. However, as soon as we try to test its extrapolation into the future, as shown in the third row, performance completely deteriorates. We test two different explanations for this behavior. First, NDP needs a context variable. This variable is usually calculated from a subset of the observations of the trajectory we would like to fit. Clearly, when we extrapolate into the future, no such observations are observed. In our experiments, we thus feed it with the past observations to extract context from. To investigate if this is the main reason for its bad performance, we provide NDP with a subset of

the future observations at prediction times. Note that this means that we provided it with more than half of the values that it should have learned to predict. Nevertheless, as shown in the fourth row, this only reduced the nll minimally. Second, we want to see if extrapolation over time leads to the problem. To evaluate this hypothesis, we decide to explicitly tell NDP about suspected, periodic behavior of the data set, i.e. we provide it at prediction with the prediction times modulo 50.

This significantly improves the test nlls. However, it also defies the main purpose of this study. Learning about the periodicity of the data is the key challenge when trying to extrapolate over time. Clearly, NDP fails to do so. Furthermore, even though we provide significant expert knowledge here, the nlls are still worse than what we obtain for the joint DGM model, even though the DGM model does not need any unfair advantages.

train	1.29 +- 0.0523
val	1.30 +- 0.103
test	295 +- 161
test, future context	277 +- 167
test, periodic times	1.58 +- 0.106

TABLE 4.6: Test likelihoods of NDP, with mean and standard deviation evaluated over the same 10 samples of 200 trajectories previously used to evaluate DGM. Again, when calculating the nlls, the third dimension is omitted.

A visualization of the same trajectory as in Figure 4.9 is shown in Figure 4.10. Here, we can speculate about the reasons for the bad performance of NDP. In the first row, it seems that the algorithm is overfitting to the noise in the training trajectories. Interestingly, despite this overfitting, the validation nlls are very decent. This is a key issue, since the main tool for regularization in NDP is early stopping using that validation nll. It seems that this tool is not enough. It remains to speculate that the structure of NDP, including the context variable that is provided with a subset of the data that should be predicted, seems to encourage some kind of overfitting.

4.4.6 *Conclusion*

In summary, we have presented a novel perspective on how to tackle gradient matching from a theoretical view. We introduced a different type of stochasticity in the dynamics model and perform probabilistic matching.

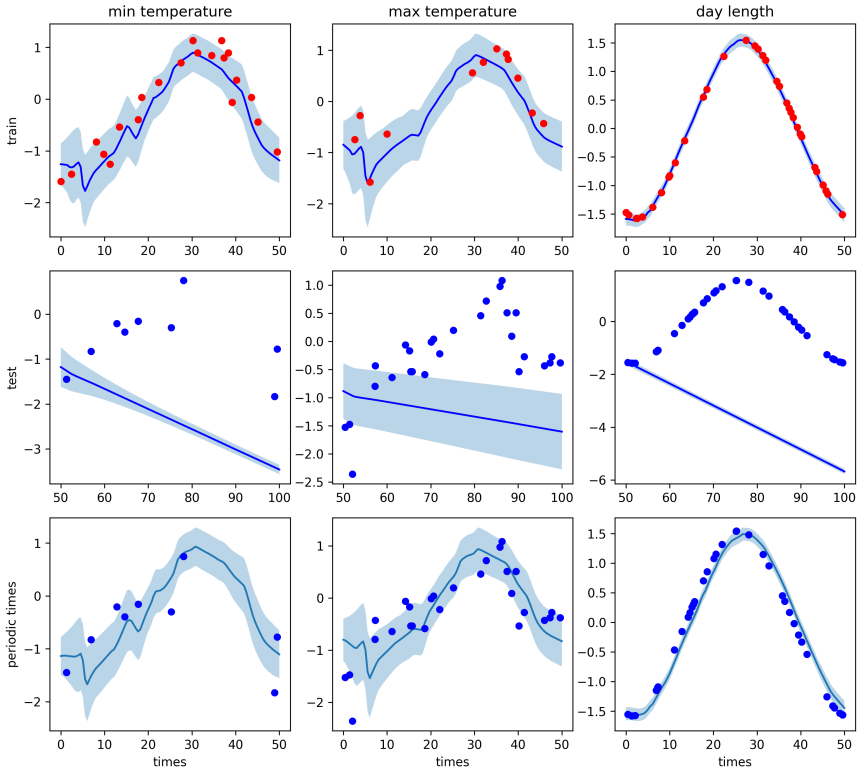


FIGURE 4.10: One sample trajectory visualizing NDP performance.

If the underlying assumption of the model are fulfilled, it can deliver impressive feats in both an interpolation and extrapolation context. If the underlying assumptions of the model are not fulfilled, it can still help significantly in the context of extrapolation, especially since comparable, uncertainty aware algorithms seem to struggle a lot with this challenge.

5

THE STOCHASTIC: ADVERSARIAL AND MMD-MINIMIZING REGRESSION

5.1 INTRODUCTION

Thus far, we exclusively considered ODEs as a model class. However, ODEs fail to incorporate the inherent stochastic behavior present in many physical, chemical or biological systems. While in principle, such effects could be captured by stochastic difference equations, these models are difficult to apply when observation times are unevenly distributed. Furthermore, they do not generalize well across observation frequencies, while natural laws do not care about this artificial construct. Thus, in this chapter, we will investigate how gradient matching can be applied in the setting of stochastic differential equations (SDEs). To this end, we use exclusively SDE models in Itô-form

$$\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\theta})\mathrm{d}t + \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{\theta})\mathrm{d}\boldsymbol{w}(t), \tag{5.1}$$

where $\boldsymbol{x}(t)$ is the time-dependent vector of states we would like to model, $\boldsymbol{\theta}$ collects the parameters of the model, $\boldsymbol{f}$ is the drift term, $\boldsymbol{g}$ is the matrix-valued diffusion function and $\boldsymbol{w}(t)$ is a Wiener-process of the same dimension as the state vector $\boldsymbol{x}$. Please note that the work in this chapter is heavily based on a conference publication, that was created jointly with several colleagues [Abb+19]. Some parts of it, including large parts of the text and all plots, have been reproduced with permission.

While SDEs can efficiently capture stochasticity and deal with unevenly spaced observation times and frequency, inference is rather challenging. Due to the stochasticity of $\boldsymbol{w}(t)$, the state vector $\boldsymbol{x}(t)$ is itself a random variable. Except for few special cases, it is not possible to find an analytic solution for the statistics of $\boldsymbol{x}(t)$ for general drift and diffusion terms. The problem is even more challenging if we were to condition on or state-estimate some discrete time observations $\boldsymbol{y}$ (filtering/smoothing) or infer some statistics for the parameters $\boldsymbol{\theta}$ (parameter inference). It is well known that the parameter inference problem is a difficult task, with most approaches either being very sensitive to initialization [Pico7], strongly dependent on the choice of hyperparameters like the spacing of the integration grid [BMR15] or using

excessive amount of computational resources even for small scale systems and state-of-the-art implementation [Ryd+18].

The difficulty of the parameter estimation problem of estimating parameters of drift and diffusion under observational noise is readily exemplified by the fact that even major scientific programming environment providers like MATLAB still lack an established toolbox for practical use.

5.1.1 *Related Work*

While it is impossible to cover all related research efforts, we would like to give a quick overview by mentioning some of the most relevant. For a more in-depth discussion, we recommend Tronarp and Särkkä [TS19], who provide an excellent review of the current state-of-the-art smoothing schemes. Moreover, Sørensen [Søro4], Nielsen, Madsen, and Young [NMY00] and Hurn, Jeisman, and Lindsay [HJL07] provide extensive explanations of the more traditional approaches.

Most classical methods rely on calculating the probability of sample paths $\boldsymbol{x}$ conditioned on the system parameters $\boldsymbol{\theta}$, denoted as $p(\boldsymbol{x}|\boldsymbol{\theta})$. Since $p(\boldsymbol{x}|\boldsymbol{\theta})$ is usually analytically intractable, approximation schemes are necessary. Elerian, Chib, and Shephard [ECS01] and Eraker [Era01] use the Euler-Maruyama discretization to approximate $p(\boldsymbol{x}|\boldsymbol{\theta})$ on a fixed, fine grid of artificial observation times later to be leveraged in a MCMC sampling scheme. Pieschner and Fuchs [PF20] and Meulen, Schauer, et al. [MS+17] subsequently refine this approach with improved bridge constructs and incorporated partial observability. Ryder et al. [Ryd+18] follow up on this idea by combining discretization procedures with variational inference. Särkkä et al. [Sär+15] investigate different approximation methods based on Kalman filtering, while Archambeau et al. [Arc+07] and Vrettas, Opper, and Cornford [VOC15] use a variational Gaussian process-based approximation. Finally, it should be mentioned that $p(\boldsymbol{x}|\boldsymbol{\theta})$ can be inferred by solving the Fokker-Planck-Kolmogorov equations using standard methods for PDEs [HL99; Ait02].

Instead of approximating $p(\boldsymbol{x}, \boldsymbol{\theta})$ in a variational fashion, Gaussian processes can as well be used to directly model $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$ and $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$, ignoring in this way any prior knowledge about their parametric form. This approach was investigated by Ruttor, Batz, and Opper [RBO13], whose linearization and discretization assumptions which were later relaxed by Yildiz et al. [Yil+18]. While we will show in our experiments that these methods can be used for parameter estimation if the parametric form of drift and diffusion

are known, it should be noted that parameter inference was not the original goal of their work.

5.1.2 *Our Work*

To the best of our knowledge, there are only very few works that try to circumvent calculating $p(\boldsymbol{x}|\boldsymbol{\theta})$ at all. Our approach is probably most closely related to the ideas presented by Riesinger, Neckel, and Rupp [RNR16]. Our proposal relies on the Doss-Sussman transformation [Dos77; Sus78] to reduce the parameter inference problem to parameter inference in an ensemble of random ordinary differential equations (RODEs). These equations can then be solved path-wise using either standard numerical schemes or using the computationally efficient gradient matching scheme of Gorbach, Bauer, and Buhmann [GBB17] as proposed by Bauer et al. [Bau+17].

The path-wise method by Bauer et al. [Bau+17] has natural parallelization properties, but there is still an inherent approximation error due to the Monte Carlo estimation of the expectation over the stochastic element in the RODE. Furthermore, their framework imposes severe linearity restrictions on the functional form of the drift $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$, while it is unable to estimate the diffusion matrix $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$.

While we will keep their assumption of a constant diffusion matrix, i.e. $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{G}$, our approach gets rid of the linearity assumptions on the drift $\boldsymbol{f}$. Furthermore, we substitute the Monte Carlo approximation by embedding the SDE into a fully statistical framework, allowing for efficient estimation of both $\boldsymbol{G}$ and $\boldsymbol{\theta}$ using state-of-the-art statistical inference methods.

Despite a constant diffusion assumption might seem restrictive at first, such SDE models are widely used, e.g. in chemical engineering [KM18], civil engineering [Jim+08], pharmacology [DS13] and of course in signal processing, control and econometrics. While we believe that this approach could be extended approximately to systems with general diffusion matrices, we leave this for future work.

In this chapter, we derive a new statistical framework for diffusion and drift parameter estimation of SDEs using the Doss-Sussmann transformation and Gaussian processes. This framework leads to a grid-free, computationally efficient and robust parameter inference scheme that combines a non-parametric Gaussian process model with adversarial loss functions. As we demonstrate in our experiments section, this inference scheme is

able to estimate constant but non-diagonal diffusion terms of stochastic differential equations without any functional form assumption on the drift. Furthermore, we show that our method significantly outperforms the state-of-the-art algorithms for SDEs with multi-modal state posteriors, both in terms of diffusion and drift parameter estimation. As with all research conducted in this thesis, the code is publicly available at `https://github.com/gabb7/AReS-MaRS`.

5.2 BACKGROUND

Following Equation (1.2) and the notation introduced in Section 1.1, we consider SDEs of the form

$$\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\theta})\mathrm{d}t + \boldsymbol{G}\mathrm{d}\boldsymbol{w}(t), \tag{5.2}$$

where $\boldsymbol{x}(t) = [\boldsymbol{x}_1(t), \ldots, \boldsymbol{x}_K(t)]^\top$ is the K-dimensional state vector at time t; $\mathrm{d}\boldsymbol{w}(t)$ are the increments of a standard Wiener process; $\boldsymbol{f}$ is an arbitrary, potentially highly nonlinear function whose parametric form is known, save for the unknown parameter vector $\boldsymbol{\theta}$; $\boldsymbol{G}$ is the unknown but constant diffusion matrix. Without loss of generality, we can assume $\boldsymbol{G}$ to be a lower-diagonal, positive semi-definite matrix.

The system is observed at N arbitrarily spaced time points

$$\boldsymbol{t} = [t_1, \ldots, t_N], \tag{5.3}$$

subjected to Gaussian observation noise:

$$\boldsymbol{y}(t_n) = \boldsymbol{x}(t_n) + \boldsymbol{e}(t_n) \qquad \forall\, n = 1, \ldots, N, \tag{5.4}$$

where we assume the noise variances to be state-dependent but time-independent, i.e.

$$p(\boldsymbol{e}(t_n)) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{e}_k(t_n) \mid 0, \sigma_k\right), \tag{5.5}$$

for $n = 1, \ldots, N$ and $k = 1, \ldots, K$.

5.2.1 *Deterministic ODE Case*

As identified by Calderhead, Girolami, and Lawrence [CGL09] and as discussed in the previous chapters, numerical integration cacn be identified as

the main culprit for the bad computational performance of Bayesian parameter inference schemes for deterministic ODEs. To circumvent numerical integration, the parameter estimation is thus turned on its head: instead of calculating $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ using numerical integration, two probabilistic estimates for the derivatives are extracted, one using only the noisy observations $\boldsymbol{y}$ and one using the differential equations. The main challenge is then to combine these two distributions, such that more information about $\boldsymbol{y}$ can guide towards better parameter estimates $\boldsymbol{\theta}$. For this purpose, Calderhead, Girolami, and Lawrence [CGL09] propose a product of experts heuristics that was accepted and reused until we demonstrated severe theoretical issues in [Wen+19], on which Chapter 2 is based on. Instead, we propose to use an alternative graphical model, forcing equality between the data based and the ODE based model save for a Gaussian distributed slack variable.

In this paper, we will provide one more interpretation of gradient matching in the Bayesian setting. Similar to the previously introduced ideas in the gradient matching context, it is aimed at finding parameters $\boldsymbol{\theta}$ such that the two distributions over $\dot{\boldsymbol{x}}$ match as closely as possible. Acknowledging the fact that standard methods like minimizing the KL divergence are not tractable, we use robust moment matching techniques while solving a much harder problem with $\boldsymbol{G} \neq \boldsymbol{0}$. However, it should be clear that our methodology could easily be applied to the special case of deterministic ODEs and thus provides an additional contribution towards parameter estimation for systems of ODEs.

5.2.2 *Notation*

Throughout this chapter, bold, capital letters describe matrices. Values of a time-dependent quantities such as the state vector $\boldsymbol{x}(t)$ can be collected in the matrix $\boldsymbol{X} = [\boldsymbol{x}(t_1), \dots, \boldsymbol{x}(t_N)]$ of dimensions $K \times N$, where the k-th row collect the N single-state values at times $\boldsymbol{t} = [t_1, \dots, t_N]$ for the state k.

The matrix $\boldsymbol{X}$ can be vectorized by concatenating its rows and defining in this way the vector $\boldsymbol{x} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_K]^\top$. This vector should not be confused with $\boldsymbol{x}(t)$, which is still a time-dependent vector of dimension K.

As we work with Gaussian processes, it is useful to standardize the state observations by subtracting the mean and dividing by the standard deviation, in a state-wise fashion. We define the vector of the data standard deviation $\boldsymbol{\sigma}_{\boldsymbol{y}} = [\sigma_{\boldsymbol{y}_1}, \dots, \sigma_{\boldsymbol{y}_K}]$, and the matrix $\boldsymbol{S}$ as:

$$\boldsymbol{S} = \boldsymbol{\sigma}_{\boldsymbol{y}} \otimes \boldsymbol{I}_N \tag{5.6}$$

where $\otimes$ indicates the Kronecker product and $\boldsymbol{I}_N$ is the identity matrix of size $N \times N$. Similarly for the means, we can define the $N \times K$ vector $\boldsymbol{\mu_y}$ that contains the K state-wise means of the observations, each repeated N times. Thus the standardize vector $\tilde{\boldsymbol{x}}$ can be defined as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{S}^{-1}(\boldsymbol{x} - \boldsymbol{\mu_y}). \tag{5.7}$$

For the sake of clarity, we omit the normalization in the following sections. It should be noted however that in a finite sample setting, standardization strongly improves the performance of GP regression. In our implementation and all the experiments in section 5.4, we assume a GP prior on the states standardized using the state-wise mean and standard deviation of the observations $\boldsymbol{y}$.

5.3 METHODS

In the deterministic case with $\boldsymbol{G} = \boldsymbol{0}$, the GP regression model can be directly applied to the states $\boldsymbol{x}$ (see Chapter 2). However, if $\boldsymbol{G} \neq \boldsymbol{0}$, the derivatives of the states with respect to time t no longer exist due to the contributions of the Wiener process. Thus, performing direct gradient matching on the states is not feasible.

5.3.1 *Latent States Representation*

We propose to tackle this problem by introducing a latent variable $\boldsymbol{z}$, defined via the linear coordinate transformation

$$\boldsymbol{z}(t) = \boldsymbol{x}(t) - \boldsymbol{o}(t), \tag{5.8}$$

where $\boldsymbol{o}(t)$ is the solution of the following SDE:

$$\mathrm{d}\boldsymbol{o}(t) = -\boldsymbol{o}(t) + \boldsymbol{G}\mathrm{d}\boldsymbol{w}(t). \tag{5.9}$$

Without loss of generality, we set $\boldsymbol{z}(0) = \boldsymbol{x}(0)$ and thus $\boldsymbol{o}(0) = 0$. While in principle the framework supports any initial condition as long as $\boldsymbol{z}(0) + \boldsymbol{o}(0) = \boldsymbol{x}(0)$, the reasons for this choice will become clear in Section 5.3.2.3.

Using Itô's formula, we obtain the following SDE for $\boldsymbol{z}$

$$\mathrm{d}\boldsymbol{z}(t) = \{\boldsymbol{f}(\boldsymbol{z}(t) + \boldsymbol{o}(t), \boldsymbol{\theta}) + \boldsymbol{o}(t)\}\,\mathrm{d}t \tag{5.10}$$

This means that for a given realization of $\boldsymbol{o}(t)$, we obtain a differentiable latent state $\boldsymbol{z}(t)$. In principle, we could sample realizations of $\boldsymbol{o}(t)$ and

solve the corresponding deterministic problems, which is equivalent to approximately marginalizing over $\boldsymbol{o}(t)$. However, it is actually possible to treat this problem statistically, completely bypassing such marginalization. We do this by creating probabilistic generative models for observations and derivatives analytically. The equations are derived in this section, while the final models are shown in Figure 5.1.

5.3.2 *Generative Model for Observations*

Let us define $\boldsymbol{e}(t)$ as the Gaussian observation error at time t. Using the matrix notation introduced in Section 5.2.2, we can write

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{E} = \boldsymbol{Z} + \boldsymbol{O} + \boldsymbol{E}, \tag{5.11}$$

where $\boldsymbol{Z}$ and $\boldsymbol{O}$ are the matrices corresponding to the lower-case variables introduced in the previous section. In contrast to standard GP regression, we have an additional noise term $\boldsymbol{O}$, which is the result of the stochastic process described by Equation (5.9). As in standard GP regression, it is possible to recover a closed form Gaussian distribution for each term.

5.3.2.1 *GP Prior*

We assume a zero-mean Gaussian prior over the latent states $\boldsymbol{z}$, whose covariance matrix is given by a kernel function $k(x, y)$, in turn parameterized by the hyperparameter vector $\boldsymbol{\phi}$:

$$p(\boldsymbol{z} \mid \boldsymbol{\phi}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}\right). \tag{5.12}$$

We treat all state dimensions as independent, meaning that we put independent GP priors with separate hyperparameters $\boldsymbol{\phi}_k$ on the time evolution of each state. Consequently, $\boldsymbol{C}_{\boldsymbol{\phi}}$ is a block diagonal matrix with K blocks each of dimension $N \times N$. The blocks model the correlation over time introduced by the GP prior.

5.3.2.2 *Error Model*

In Equation (5.5), we assume that observational errors are i.i.d. Gaussians uncorrelated over time. The joint distribution of all errors is thus still a Gaussian distribution, whose covariance $\boldsymbol{T}$ has only diagonal elements given by the GP likelihood variances $\boldsymbol{\sigma} = \{\sigma_k^2\}_{k=1}^K$. More precisely:

$$\boldsymbol{T} = \boldsymbol{\sigma} \otimes \boldsymbol{I}_N. \tag{5.13}$$

and

$$p(\boldsymbol{e} \mid \boldsymbol{\sigma}) = \mathcal{N}(\boldsymbol{e} \mid \mathbf{0}, \boldsymbol{T}). \tag{5.14}$$

5.3.2.3 *Ornstein-Uhlenbeck Process*

Through the coordinate transformation in Equation (5.8), all stochasticity is captured by the stochastic process $\boldsymbol{o}(t)$ described by Equation (5.9). Such mathematical construct has a closed-form, Gaussian solution and is called Ornstein-Uhlenbeck process. For the one-dimensional case with zero initial condition and unit diffusion

$$\mathrm{d}\hat{o}(t) = -\hat{o}(t) + \mathrm{d}w(t), \tag{5.15}$$

we get the following mean and covariance:

$$\mathbb{E}[\hat{o}(t)] = 0 \tag{5.16}$$

$$\mathrm{cov}[\hat{o}(t_i), \hat{o}(t_j)] = \frac{1}{2} e^{-|t_i - t_j|} - \frac{1}{2} e^{-(t_i + t_j)}. \tag{5.17}$$

Sampling $\hat{o}(t)$ at the N points $\boldsymbol{t} = [t_1, \ldots, t_N]$ yields the vector $\hat{\boldsymbol{o}}(\boldsymbol{t}) = [\hat{o}(t_1), \ldots, \hat{o}(t_N)]$, which is Gaussian distributed:

$$p(\hat{\boldsymbol{o}}(\boldsymbol{t})) = \mathcal{N}(\hat{\boldsymbol{o}}(\boldsymbol{t}) \mid \mathbf{0}, \boldsymbol{\Omega}_{\text{one}}), \tag{5.18}$$

where $[\boldsymbol{\Omega}_{\text{one}}]_{ij} = \mathrm{cov}[\hat{o}(t_i), \hat{o}(t_j)]$ according to (5.17). In the case of a K-dimensional process with identity diffusion, i.e.

$$\mathrm{d}\hat{\boldsymbol{o}}(t) = -\hat{\boldsymbol{o}} + \boldsymbol{I}_K \mathrm{d}\boldsymbol{w}(t), \tag{5.19}$$

we can just treat each state dimension as an independent, one-dimensional OU process. Thus, after sampling $\hat{\boldsymbol{o}}(t)$ K times at the N time points in $\boldsymbol{t}$ and unrolling the resulting matrix as described in section 5.2.2, we get

$$p(\hat{\boldsymbol{o}}) = \mathcal{N}(\hat{\boldsymbol{o}} \mid \mathbf{0}, \boldsymbol{\Omega}), \tag{5.20}$$

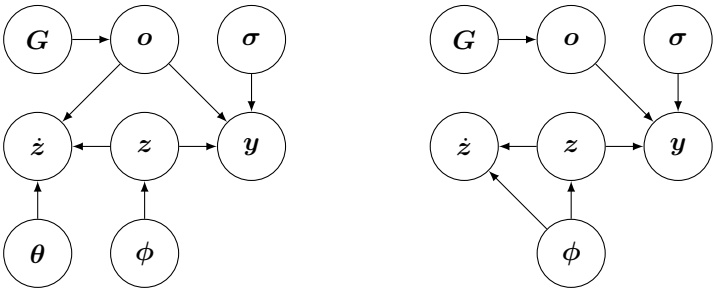
where $\boldsymbol{\Omega}$ is a block diagonal matrix with one $\boldsymbol{\Omega}_{\text{one}}$ for each state dimension.

Using Itô's formula, we can show that the samples of the original Ornstein-Uhlenbeck process $\boldsymbol{o}$ at each time point can be obtained via the linear coordinate transformation

$$\boldsymbol{o}(t) = \boldsymbol{G}\hat{\boldsymbol{o}}(t). \tag{5.21}$$

Let $\boldsymbol{B}$ be defined as the matrix that performs this linear transformation for the unrolled vectors $\boldsymbol{o} = \boldsymbol{B}\hat{\boldsymbol{o}}$. We can then write the density of the original OU process as

$$p(\boldsymbol{o} \mid \boldsymbol{G}) = \mathcal{N}\left(\boldsymbol{o} \mid \mathbf{0}, \boldsymbol{B}\boldsymbol{\Omega}\boldsymbol{B}^\top\right). \tag{5.22}$$



(a) SDE-based model (b) Data-based model

FIGURE 5.1: Generative models for the two different ways to compute the derivatives of the latent states $\boldsymbol{z}$.

5.3.2.4 *Marginals of the Observations*

Using Equation (5.11), the marginal distribution of $\boldsymbol{y}$ can be computed as the sum of three independent Gaussian-distributed random variables with zero mean, described respectively by Equations (5.12), (5.14) and (5.22). Thus, $\boldsymbol{y}$ is again Gaussian-distributed, according to

$$p(\tilde{\boldsymbol{y}} \mid \boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = \mathcal{N}(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{\Sigma}), \tag{5.23}$$

where

$$\boldsymbol{\Sigma} = \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T} + \boldsymbol{B}\boldsymbol{\Omega}\boldsymbol{B}^T. \tag{5.24}$$

Thanks to the latent state representation, the diffusion matrix $\boldsymbol{G}$ is now a part of the hyperparameters of the observation model. It can then be inferred alongside the hyperparameters of the GP using maximum evidence [Raso4c]. Using a stationary kernel k, $\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T}$ captures the stationary part of $\boldsymbol{z}$ as in standard GP regression, while the parameters in $\boldsymbol{G}$ describe the non-stationary part due deriving from $\boldsymbol{\Omega}$. This ultimately leads to an identifiable problem.

5.3.3 *Generative Model for Derivatives*

Similarly to gradient matching approaches, we define two generative models for the derivatives $\dot{\boldsymbol{z}}$, one based on the data and one based on the SDE model.

5.3.3.1 *Data-based Model*

As demonstrated e.g. in Chapter 2, the prior defined in Equation (5.12) automatically induces a GP prior on the conditional derivatives of $\boldsymbol{z}$. Defining

$$\boldsymbol{D} \coloneqq {}'\boldsymbol{C}_{\boldsymbol{\phi}} \boldsymbol{C}_{\boldsymbol{\phi}}^{-1}, \tag{5.25}$$

$$\boldsymbol{A} \coloneqq \boldsymbol{C}''_{\boldsymbol{\phi}} - {}'\boldsymbol{C}_{\boldsymbol{\phi}} \boldsymbol{C}_{\boldsymbol{\phi}}^{-1} \boldsymbol{C}'_{\boldsymbol{\phi}} \tag{5.26}$$

where

$$\left[{}'\boldsymbol{C}_{\boldsymbol{\phi}}\right]_{i,j} \coloneqq \frac{\partial}{\partial a} k_{\boldsymbol{\phi}}(a,b)\Big|_{a=t_i, b=t_j}, \tag{5.27}$$

$$\left[\boldsymbol{C}'_{\boldsymbol{\phi}}\right]_{i,j} \coloneqq \frac{\partial}{\partial b} k_{\boldsymbol{\phi}}(a,b)\Big|_{a=t_i, b=t_j}, \tag{5.28}$$

$$\left[\boldsymbol{C}''_{\boldsymbol{\phi}}\right]_{i,j} \coloneqq \frac{\partial^2}{\partial a \partial b} k_{\boldsymbol{\phi}}(a,b)\Big|_{a=t_i, b=t_j}, \tag{5.29}$$

we can write

$$p(\dot{\boldsymbol{z}} \mid \boldsymbol{z}, \boldsymbol{\phi}) = \mathcal{N}\left(\dot{\boldsymbol{z}} \mid \boldsymbol{D}\boldsymbol{z}, \boldsymbol{A}\right). \tag{5.30}$$

5.3.3.2 *SDE-based Model*

There also is a second way of obtaining an expression for the derivatives of $\boldsymbol{z}$, namely using Equation (5.10):

$$p(\dot{\boldsymbol{z}} \mid \boldsymbol{o}, \boldsymbol{z}, \boldsymbol{\theta}) = \delta(\dot{\boldsymbol{z}} - \boldsymbol{f}(\boldsymbol{z} + \boldsymbol{o}, \boldsymbol{\theta}) - \boldsymbol{o}), \tag{5.31}$$

where δ represents the dirac delta.

5.3.4 *Inference*

Combined with the modeling paradigms introduced in the previous sections, this yields the two generative models for the observations in Figure 5.1. The graphical model in Figure 5.1a represents the derivatives we get via the generative process described by the SDEs, in particular the nonlinear drift function $\boldsymbol{f}$. The model in Figure 5.1b represents the derivatives yielded by the generative process described by the Gaussian process. Assuming a perfect GP fit and access to the true parameters $\boldsymbol{\theta}$, intuitively these two distributions should be equal. We thus want to find parameters $\boldsymbol{\theta}$ that minimizes the difference between these two distributions.

Compared to the deterministic ODE case, the graphical models in Figure 5.1 contain additional dependencies on the contribution of the OU process $\boldsymbol{o}$. Furthermore, the SDE-driven probability distribution of $\dot{\boldsymbol{z}}$ in Figure 5.1a depends on $\boldsymbol{z}$, $\boldsymbol{o}$, and $\boldsymbol{\theta}$ through a potentially highly nonlinear drift function $\boldsymbol{f}$. Thus, one cannot do analytical inference without making restrictive assumptions on the functional form of $\boldsymbol{f}$.

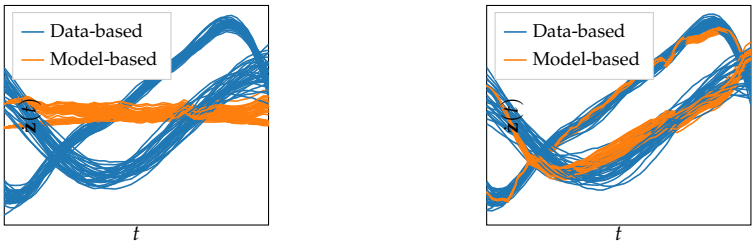
However, as shown in Section A.4.3 of the appendix , it is possible to derive computationally efficient ancestral sampling schemes for both models, as summarized in Algorithm 3. While this rules out classical approaches like analytically minimizing the KL divergence, we can now deploy likelihood-free algorithms that were designed for matching two probability densities based on samples.

Algorithm 3 Ancestral sampling for $\dot{\boldsymbol{z}}$

Input: $\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{z}, \boldsymbol{\theta}), \boldsymbol{t}, \boldsymbol{\sigma}, \boldsymbol{G}$
Ancestral sampling the SDE model
Sample $\boldsymbol{o}_\mathrm{s}$ by drawing from $p(\boldsymbol{o} \mid \boldsymbol{G})$
Sample $\boldsymbol{z}_\mathrm{s}$ by drawing from $p(\boldsymbol{z} \mid \boldsymbol{y}, \boldsymbol{o}, \boldsymbol{\sigma})$ using $\boldsymbol{o}_\mathrm{s}$
Sample $\dot{\boldsymbol{z}}_\mathrm{s}$ by drawing from $p(\dot{\boldsymbol{z}} \mid \boldsymbol{o}, \boldsymbol{z}, \boldsymbol{\theta})$ using $\boldsymbol{o}_\mathrm{s}, \boldsymbol{z}_\mathrm{s}$
Ancestral sampling the Data model
Sample $\boldsymbol{z}_\mathrm{d}$ by drawing from $p(\boldsymbol{z} \mid \boldsymbol{y}, \boldsymbol{G}, \boldsymbol{\sigma})$
Sample $\dot{\boldsymbol{z}}_\mathrm{d}$ by drawing from $p(\dot{\boldsymbol{z}} \mid \boldsymbol{z}, \boldsymbol{\phi})$
Return: $\dot{\boldsymbol{z}}_\mathrm{s}, \dot{\boldsymbol{z}}_\mathrm{d}$

5.3.5 *Adversarial Sample-based Inference*

Arguably, generative adversarial networks (GANs) [Goo+14] are amongst the most popular algorithms of this kind; here a parametric neural network is trained to match the unknown likelihood of the data. The basic GAN setup consists of a fixed data set, a generator that tries to create realistic samples of said dataset and a discriminator that tries to tell apart the fake samples from the true ones. As recently shown by Yang, Zhang, and Karniadakis [YZK20], GANs have the potential to solve stochastic partial differential equations (SPDEs). Yang, Zhang, and Karniadakis [YZK20] assume a fixed data set consisting of observations (similar to the $\boldsymbol{y}$ in this paper) and use an SPDE-inspired neural network as a generator for realistic observations. In the case of SDEs however, this would still involve a lot of numerical integration. Thus, we modify the GAN setup by leaving behind



(a) Before Training

(b) After Training

FIGURE 5.2: Comparing gradients sampled from the graphical model in Figure 5.1a (Model-based) and the graphical model in Figure 5.1b (Data-based) before and after adversarial training on Lotka Volterra.

the idea of having a fixed data set. Instead of relying on bootstrapped samples of the observations $\boldsymbol{y}$, we sample the derivatives from the data-based model shown in Figure 5.1b. For a sufficiently good model fit, these samples represent the true derivatives of the latent variable $\boldsymbol{z}$. We then use the SDE-based model shown in Figure 5.1a as a generator. To avoid standard GAN problems such as training instability and to improve robustness, we choose to replace the discriminator with a critic $\mathcal{C}_\omega$. As proposed by Arjovsky, Chintala, and Bottou [ACB17], this critic is trained to estimate the Wasserstein distance between the derivative samples. The resulting algorithm, summarized in Algorithm 4, can be interpreted as performing Adversarial Regression for SDEs and will thus be called AReS. In Figure 5.2, we show the derivatives sampled from the two models both before and after training for one example run of the Lotka Volterra system (cf. Section 5.4.4). While not perfect, the GAN is clearly able to push the SDE gradients towards the gradients of the observed data.

5.3.6 *Maximum Mean Discrepancy*

Even though they work well in practical settings, during training GANs need ad hoc precautions and careful balancing between their generator and discriminator. Dziugaite, Roy, and Ghahramani [DRG15] propose to solve this problem using Maximum Mean Discrepancy (MMD) [Gre+12] as a metric to substitute the discriminator. As proposed by Li, Swersky, and Zemel [LSZ15], we choose the rational quadratic kernel to obtain a robust discriminator that can be deployed without fine-tuning on a variety

of problems. The resulting procedure, summarized in Algorithm 5, can be interpreted as performing Maximum mean discrepancy-minimizing Regression for SDEs and will thus be called MaRS.

Algorithm 4 AReS

Input: Observations $\boldsymbol{y}$ at times $\boldsymbol{t}$, a model $\boldsymbol{f}$, learning rate α, number of total iterations N_{it}, the clipping parameter c, the batch size M, the number of iterations of the critic per generator iteration n_{critic}.
Train the Gaussian process on the data to recover the hyperparameters ϕ, $\boldsymbol{\sigma}$ and the diffusion $\boldsymbol{G}$
Initialize the critic parameters $\boldsymbol{\omega}$ and the SDE parameters $\boldsymbol{\theta}$ respectively with $\boldsymbol{\omega}_0$ and $\boldsymbol{\theta}_0$
for $n_{\text{it}} = 1, \dots, N_{\text{it}}$ **do**
    for $n_{\text{c}} = 1, \dots, n_{\text{critic}}$ **do**
        Sample $\dot{\boldsymbol{z}}_{\text{s}} \sim p_{\text{s}}(\dot{\boldsymbol{z}})$ and $\dot{\boldsymbol{z}}_{\text{d}} \sim p_{\text{d}}(\dot{\boldsymbol{z}})$ as described in Algorithm 3. Each batch contains M elements
        $g_{\boldsymbol{\omega}} \leftarrow \nabla_{\boldsymbol{\omega}} \left[\frac{1}{M} \sum_{i=1}^{M} C_{\boldsymbol{\omega}}(\dot{\boldsymbol{z}}_{\text{d}}^{(i)}) - \frac{1}{M} \sum_{i=1}^{M} C_{\boldsymbol{\omega}}(\dot{\boldsymbol{z}}_{\text{s}}^{(i)}) \right]$
        $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \alpha \cdot \text{Adam}(\boldsymbol{\omega}, g_{\boldsymbol{\omega}})$
        $\boldsymbol{\omega} \leftarrow \text{clip}(\boldsymbol{\omega}, -c, c)$
    end for
    $g_{\boldsymbol{\theta}} \leftarrow -\nabla_{\boldsymbol{\theta}} \frac{1}{M} \sum_{i=1}^{M} f_{\boldsymbol{\omega}}(\dot{\boldsymbol{z}}_{\text{s}}^{(i)})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \text{Adam}(\boldsymbol{\theta}, g_{\boldsymbol{\theta}})$
end for

Algorithm 5 MaRS

Input: Observations $\boldsymbol{y}$ at times $\boldsymbol{t}$, SDE model $\boldsymbol{f}$, learning rate α, number of iterations N_{it}, batch size M
Train the Gaussian process on the data to recover the hyperparameters ϕ, $\boldsymbol{\sigma}$ and the diffusion $\boldsymbol{G}$
Initialize the SDE parameters with $\boldsymbol{\theta}_0$
for $n_{\text{it}} = 1, \dots, N_{\text{it}}$ **do**
    Sample $\dot{\boldsymbol{z}}_{\text{s}} \sim p_{\text{s}}(\dot{\boldsymbol{z}})$ and $\dot{\boldsymbol{z}}_{\text{d}} \sim p_{\text{d}}(\dot{\boldsymbol{z}})$ as described in Algorithm 3. Each batch contains M elements
    $g_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}} MMD_u^2 [\dot{\boldsymbol{z}}_{\text{s}}, \dot{\boldsymbol{z}}_{\text{d}}]$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \text{Adam}(\boldsymbol{\theta}, g_{\boldsymbol{\theta}})$
end for

5.4 EXPERIMENTS

To evaluate the empirical performance of our method, we conduct several experiments on simulated data, using four standard benchmark systems and comparing against the EKF-based approach by Särkkä et al. [Sär+15] and two GP-based approaches respectively by Vrettas, Opper, and Cornford [VOC15] and Yildiz et al. [Yil+18]. While the drift of some systems is equivalent to systems previously used as ODE-benchmarks, the systems in this section also include a diffusion term. This makes them completely different, both in behavior as well as in its requirements on a parameter inference algorithm.

5.4.1 *Setups*

The first system is a simple Ornstein-Uhlenbeck process as shown in Figure 5.3a, given by the SDE

$$\mathrm{d}x(t) = \theta_0(\theta_1 - x(t))\mathrm{d}t + G\mathrm{d}w(t). \tag{5.32}$$

As mentioned in Section 5.3.2.3, this system has an analytical Gaussian process solution and thus serves more academic purposes. We use $\boldsymbol{\theta} = [0.5, 1.0]$, $G = 0.5$ and $x(0) = 10$.

The second system is the Lorenz '63 model given by the SDEs

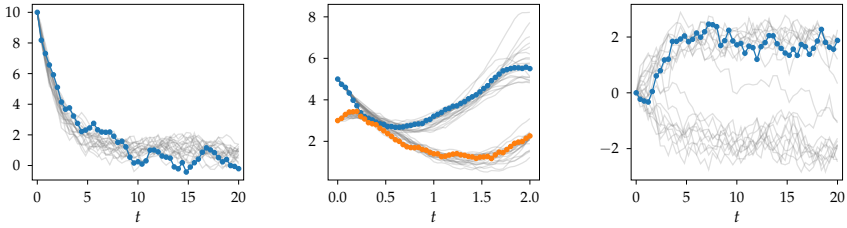
$$\begin{aligned}
\mathrm{d}x_1(t) &= \theta_1(x_2(t) - x_1(t))\mathrm{d}t &&+ \sigma_1\mathrm{d}w_1(t)\\
\mathrm{d}x_2(t) &= (\theta_2 x_1(t) - x_2(t) - x_1(t)x_3(t))\mathrm{d}t &&+ \sigma_2\mathrm{d}w_2(t)\\
\mathrm{d}x_3(t) &= (x_1(t)x_2(t) - \theta_3 x_3(t))\mathrm{d}t &&+ \sigma_3\mathrm{d}w_3(t).
\end{aligned}$$

In both systems, the drift function is linear in one state or one parameter conditioned on all the others [cf. GBB17]. Furthermore, there is no coupling across state dimensions in the diffusion matrix. This leads to two more interesting test cases.

To investigate the algorithm's capability to deal with off-diagonal terms in the diffusion, we introduce the two dimensional Lotka-Volterra system shown in Figure 5.3b, given by the SDEs

$$\mathrm{d}\boldsymbol{x}(t) = \begin{bmatrix} \theta_1 x_1(t) - \theta_2 x_1(t)x_2(t) \\ -\theta_3 x_2(t) + \theta_4 x_1(t)x_2(t) \end{bmatrix} \mathrm{d}t + \boldsymbol{G}\mathrm{d}\boldsymbol{w}(t), \tag{5.33}$$

where $\boldsymbol{G}$ is, without loss of generality, assumed to be a lower triangular matrix. The true vector parameter is $\boldsymbol{\theta} = [2, 1, 4, 1]$ and the system is



(a) Ornstein Uhlenbeck (b) Lotka Volterra (c) Double Well

FIGURE 5.3: Sample trajectories for three different benchmark systems. While Ornstein Uhlenbeck and Lotka Volterra are rather tame, the Double Well potential clearly exhibits a bifurcation effect.

simulated starting from $\boldsymbol{x}(0) = [3, 5]$. Since its original introduction by Lotka [Lot32], the Lotka Volterra system has been widely used to model population dynamics in biology. The system is observed at 50 equidistant points in the interval $t = [0,\ 20]$. As it turns out, this problem is significantly challenging for all algorithms, despite the absence of observation noise.

To investigate the effect of strong non-linearities in the drift, we introduce the Ginzburg-Landau double-well potential shown in Figure 5.3c, defined by the SDE

$$\mathrm{d}x(t) = \theta_0 x(\theta_1 - x^2)\mathrm{d}t + G\mathrm{d}w(t). \tag{5.34}$$

Using $\boldsymbol{\theta} = [0.1, 4]$, $G = 0.5$ and $x(0) = 0$, this system exhibits an interesting bifurcation effect. While there are two stable equilibria at $x = \pm 2$, the one the system will end up in is completely up to noise. For this reason it represents a fitting framework to test how well an algorithm can deal with multi-modal SDEs. The potential value is observed at 50 equidistant points in the interval $t = [0,\ 20]$, subjected to observational noise with $\sigma = 0.2$.

Lastly, some implementation details are constant throughout each experiment: the critic in the adversarial parameter estimation is a 2-layer fully connected neural network, with respectively 256 and 128 nodes. Every batch, for both MMD and adversarial training contains 256 elements. While the Ornstein-Uhlenbeck process and the double-well potential were modeled with a sigmoid kernel, for Lotka-Volterra and Lorenz '63 we used a common RBF (we point at Rasmussen [Ras04c] for more information about kernels and GPs).

5.4.2 *Evaluation*

For all systems, the parameters $\boldsymbol{\theta}$ turn out to be identifiable. Thus, the parameter value is a good indicator of how well an algorithm is able to infer the drift function. However, since the components of $\mathrm{d}\boldsymbol{w}(t)$ are independent, there are multiple diffusion matrices $\boldsymbol{G}$ that generate the same trajectories. We thus directly compare the variance of the increments, i.e. the elements of $\boldsymbol{H} := \boldsymbol{G}^T\boldsymbol{G}$.

To account for statistical fluctuations, we use 100 independent realizations of the SDE systems and compare the mean and standard deviation of $\boldsymbol{\theta}$ and $\boldsymbol{H}$. AReS and MaRS are compared against the Gaussian process-based VGPA by Vrettas, Opper, and Cornford [VOC15] and NPSDE by Yildiz et al. [Yil+18] as well as the classic Kalman filter-based ESGF recommended by Särkkä et al. [Sär+15].

5.4.3 *Locally Linear Systems*

As mentioned in Section 5.4.1, the functional form of the drift functions of both the Ornstein-Uhlenbeck process and the Lorenz '63 system satisfies a local linearity assumption, while their diffusion is kept diagonal. Thus, they serve as excellent benchmarks for parameter inference algorithms. The empirical results are shown in Table 5.2a for the Ornstein-Uhlenbeck. Unfortunately, VGPA turns out to be rather unstable if both diffusion and parameters are unknown, despite on average roughly 54 hours are needed to observe convergence. We then provide it with the true $\boldsymbol{G}$ and show only its empirical parameter estimates. Since both AReS and MaRS use Equation (5.23) to determine $\boldsymbol{G}$, they share the same values. Due to space restrictions, the results for Lorenz '63 can be found in Table A.4 of the appendix. As demonstrated by this experiment, AReS and MaRS can deal with locally linear systems, outperforming their competitors, especially in their estimates of the diffusion terms.

5.4.4 *Non-Diagonal Diffusion*

To investigate the effect of off-diagonal entries in $\boldsymbol{G}$, we use the Lotka-Volterra dynamics. Since NPSDE is unable to model non-diagonal diffusions, we provide it with the true $\boldsymbol{G}$ and only compare parameter estimates. As VGPA is already struggling in the lower dimensional cases, we omit it from this comparison due to limited computational resources. The results

are shown in Table 5.2b. AReS and MaRS clearly outperform the other methods in terms of diffusion estimation, while ESGF is the only algorithm that yields drift parameter estimates of comparable quality.

5.4.5 *Dealing with Multi-Modality*

As a final challenge, we investigate the Ginzburg-Landau double well potential. Despite one-dimensional, its state distribution is multi-modal even if all parameters are known. As shown in Table 5.2c, this is definitely a challenge for all classical approaches. While the number of data-points is probably not enough for the non-parametric proxy for the drift function in NPSDE, the time-dependent Gaussianity assumptions in both VGPA and ESGF are problematic in this case. In our gradient matching framework, no such assumption is made. Thus, both AReS and MaRS are able to deal with the multimodality of the problem.

Ground truth	NPSDE	VGPA	ESGF	AReS	MaRS
$\theta_0 = 0.5$	0.41 ± 0.11	0.53 ± 0.08	0.49 ± 0.07	$\mathbf{0.50 \pm 0.21}$	0.46 ± 0.06
$\theta_1 = 1$	0.71 ± 1.34	0.96 ± 0.31	0.96 ± 0.24	1.06 ± 0.93	$\mathbf{0.99 \pm 0.25}$
$H = 0.25$	0.00 ± 0.01	/	0.19 ± 0.06	$\mathbf{0.24 \pm 0.09}$	

(a) Ornstein-Uhlenbeck process

Ground truth	NPSDE	ESGF	AReS	MaRS
$\theta_0 = 2$	1.58 ± 0.71	2.04 ± 0.09	2.36 ± 0.18	$\mathbf{2.00 \pm 0.09}$
$\theta_1 = 1$	0.74 ± 0.31	1.02 ± 0.05	1.18 ± 0.9	$\mathbf{1.00 \pm 0.04}$
$\theta_2 = 4$	2.26 ± 1.51	3.87 ± 0.59	$\mathbf{3.97 \pm 0.63}$	3.70 ± 0.51
$\theta_3 = 1$	0.49 ± 0.35	0.96 ± 0.14	$\mathbf{0.98 \pm 0.18}$	0.91 ± 0.14
$\boldsymbol{H}_{1,1} = 0.05$	/	0.01 ± 0.03	$\mathbf{0.03 \pm 0.004}$	
$\boldsymbol{H}_{1,2} = 0.03$	/	0.01 ± 0.01	$\mathbf{0.02 \pm 0.01}$	
$\boldsymbol{H}_{2,1} = 0.03$	/	0.01 ± 0.01	$\mathbf{0.02 \pm 0.01}$	
$\boldsymbol{H}_{2,2} = 0.09$	/	0.03 ± 0.02	$\mathbf{0.09 \pm 0.03}$	

(b) Lotka-Volterra

Ground truth	NPSDE	VGPA	ESGF	AReS	MaRS
$\theta_0 = 0.1$	0.09 ± 7.00	0.05 ± 0.04	0.01 ± 0.03	0.09 ± 0.04	$\mathbf{0.10 \pm 0.05}$
$\theta_1 = 4$	3.36 ± 248.82	1.11 ± 0.66	0.11 ± 0.16	3.68 ± 1.34	$\mathbf{3.85 \pm 1.10}$
$H = 0.25$	0.00 ± 0.02	/	0.20 ± 0.05	$\mathbf{0.21 \pm 0.09}$	

(c) Double-Well potential

TABLE 5.1: Inferred parameters over 100 independent realizations of respectively the Ornstein-Uhlenbeck, Ginzburg-Landau Double-Well and Lotka-Volterra dynamics. For every algorithm, we show the median $\pm$ one standard deviation.

6

SUMMARY

In this thesis, we developed several novel algorithms grounded on different theoretical view points for parameter inference in time-continuous dynamical systems.

In Chapter 2, we gave a short overview of the state of the art gradient matching schemes and analyzed their theoretical shortcomings. After providing a new theoretical foundation, we demonstrated that with proper hyperparameter handling, gradient matching algorithms can perform well on simulated data sets. Crucially, our algorithm did not rely on hand-tuned hyperparameters or complicated, opaque sampling schemes, which was common in comparison methods.

In Chapter 3, we then provided a constrained-based interpretation of the theoretical framework provided in the previous chapter. Reformulating the problem as a constrained optimization model allowed us to develop significantly more accurate and faster inference schemes, albeit at the cost of properly handling uncertainty. Nevertheless, the resulting inference scheme allows to perform efficient model selection, by leveraging the model mismatch factor γ. In the context of optimization, we were also able to develop an efficient feature approximation scheme, that allows us to give guarantees on the approximation error introduced via the features in terms of the loss function.

In Chapter 4, the dynamics models become much more complex, reflecting the reality that parametric models might not always be available. In the previous chapters, the matching of gradients was achieved via a hard loss function. Here, we focus on obtaining distributions over gradients and then match them on the level of distributions. After combining the resulting algorithm with efficient approximation schemes, we were able to apply it to a data set with real world measurements. The resulting algorithm is a hybrid combination of gradient matching and numerical integration, without sacrificing the computational efficiency of gradient matching. As we demonstrated, it performs well on extrapolation tasks, even though its underlying assumptions are not fulfilled.

Finally, in Chapter 5, we apply the notion of distribution matching to the case of stochastic differential equations. Sample paths of SDEs are not differentiable with probability 1 everywhere. Nevertheless, through careful

restriction of the model class and distributional matching on the level of trajectory, we managed to develop a gradient matching scheme for this case as well.

In summary, this thesis started with a family of tuning intensive algorithms. Then, through careful innovations, we ultimately created an algorithm that can be applied to real measurements with no hyperparameter tuning necessary. As demonstrated in Chapter 4, the resulting parameter inference scheme can be applied in settings where other inference mechanisms fail. Thus, it could be argued that they are an important first step in making Bayesian parameter inference schemes practically viable for real world, time-continuous learning scenarios. It is an interesting question how and if one could leverage the uncertainties our algorithms provide for practically relevant applications, especially in the context of time-continuous reinforcement learning, in the context of providing safety guarantees, or in the context of model selection. In particular, it remains to be studied what benefits they would bring. Unfortunately, such explorations were outside of the scope of this thesis and we are excited to see what tangents will be explored in future work.

A

APPENDIX

A.1 APPENDIX TO FGPGM

A.1.1 *Proof of theorem 1*

Proof 1 *The proof of this statement follows directly by combining all the previous definitions and marginalizing out all the random variables that are not part of the end result.*

First, one starts with the joint density over all variables as stated in equation (2.19)

$$p(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}, \boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{\theta} | \boldsymbol{\phi}, \sigma, \gamma) = p_{GP}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y} | \boldsymbol{\phi}, \sigma) p_{ODE}(\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{\theta} | \boldsymbol{x}, \dot{\boldsymbol{x}}, \gamma),$$

where

$$p_{GP}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y} | \boldsymbol{\phi}, \sigma) = p(\boldsymbol{x} | \boldsymbol{\phi}) p(\dot{\boldsymbol{x}} | \boldsymbol{x}, \boldsymbol{\phi}) p(\boldsymbol{y} | \boldsymbol{x}, \sigma)$$

and

$$p_{ODE}(\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{\theta} | \boldsymbol{x}, \dot{\boldsymbol{x}}, \gamma) = p(\boldsymbol{\theta}) p(\boldsymbol{F_1} | \boldsymbol{\theta}, \boldsymbol{x}) p(\boldsymbol{F_2} | \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}) \delta(\boldsymbol{F_1} - \boldsymbol{F_2}).$$

To simplify this formula, p_{ODE} can be reduced by marginalizing out $\boldsymbol{F_2}$ using the properties of the Dirac delta function and the probability density defined in equation (2.18)*. The new p_{ODE} is then independent of $\boldsymbol{F_2}$.*

$$p_{ODE}(\boldsymbol{F_1}, \boldsymbol{\theta} | \boldsymbol{x}, \dot{\boldsymbol{x}}, \gamma) = p(\boldsymbol{\theta}) p(\boldsymbol{F_1} | \boldsymbol{\theta}, \boldsymbol{x}) \mathcal{N}(\boldsymbol{F_1} | \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}).$$

Inserting equation (2.17) *yields*

$$p_{ODE}(\boldsymbol{F_1}, \boldsymbol{\theta} | \boldsymbol{x}, \dot{\boldsymbol{x}}, \gamma) = p(\boldsymbol{\theta}) \delta(\boldsymbol{F_1} - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})) \mathcal{N}(\boldsymbol{F_1} | \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}).$$

Again, the properties of the Dirac delta function are used to marginalize out $\boldsymbol{F_1}$. The new p_{ODE} is now independent of $\boldsymbol{F_1}$,

$$p_{ODE}(\boldsymbol{\theta} | \boldsymbol{x}, \dot{\boldsymbol{x}}, \gamma) = p(\boldsymbol{\theta}) \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) | \dot{\boldsymbol{x}}, \gamma \boldsymbol{I}).$$

This reduced p_{ODE} is now combined with p_{GP}. Observing that the mean and the argument of a normal density are interchangeable and inserting the definition of the GP prior on the derivatives given by equation (2.9) leads to

$$p(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{y},\boldsymbol{\theta}|\boldsymbol{\phi},\sigma,\gamma) = p(\boldsymbol{\theta})p(\boldsymbol{x}|\boldsymbol{\phi})\mathcal{N}(\dot{\boldsymbol{x}}|\boldsymbol{D}\boldsymbol{x},\boldsymbol{A})p(\boldsymbol{y}|\boldsymbol{x},\sigma)\mathcal{N}(\dot{\boldsymbol{x}}|\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta}),\gamma\boldsymbol{I}).$$

$\dot{\boldsymbol{x}}$ can now be marginalized by observing that the product of two normal densities of the same variable is again a normal density. The formula can be found, e.g., in Petersen, Pedersen, et al. [PP+08]. As a result, one obtains

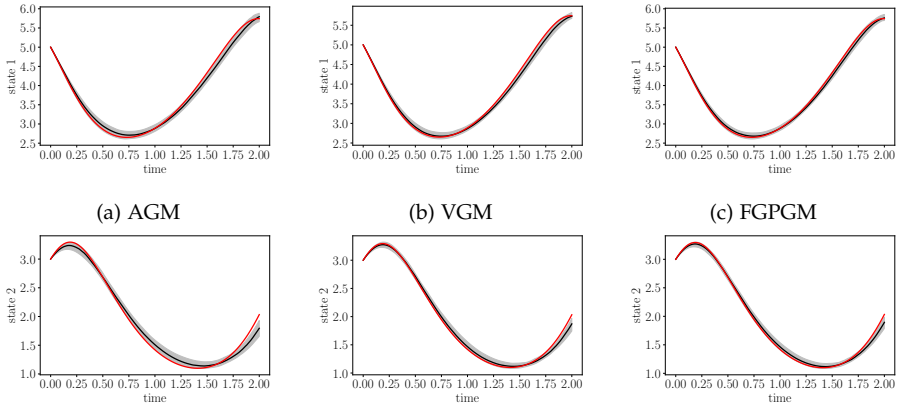
$$p(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta}|\boldsymbol{\phi},\sigma,\gamma) = p(\boldsymbol{\theta})p(\boldsymbol{x}|\boldsymbol{\phi})p(\boldsymbol{y}|\boldsymbol{x},\sigma)\mathcal{N}(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})|\boldsymbol{D}\boldsymbol{x},\boldsymbol{A}+\gamma\boldsymbol{I}).$$

It should now be clear that after inserting equations (2.7) and (2.8) and renormalizing, we get the final result

$$p(\boldsymbol{x},\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{\phi},\gamma,\sigma) \propto p(\boldsymbol{\theta})\mathcal{N}(\boldsymbol{x}|\boldsymbol{0},\boldsymbol{C}_{\boldsymbol{\phi}})\mathcal{N}(\boldsymbol{y}|\boldsymbol{x},\sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})|\boldsymbol{D}\boldsymbol{x},\boldsymbol{A}+\gamma\boldsymbol{I}),$$

concluding the proof of this theorem.

A.1.2 *Additional Plots*



(a) AGM (b) VGM (c) FGPGM

FIGURE A.1: Median Plots for all states of Lotka Volterra with low noise. State 1 is in the top row, state 2 is in the bottom row. The red line is the ground truth, while the black line and the shaded area denote the median and the 75% quantiles of the results of 100 independent noise realizations. As was already to be expected by the parameter estimates, FGPGM and VGM are almost indistinguishable while AGM falls off a little bit.

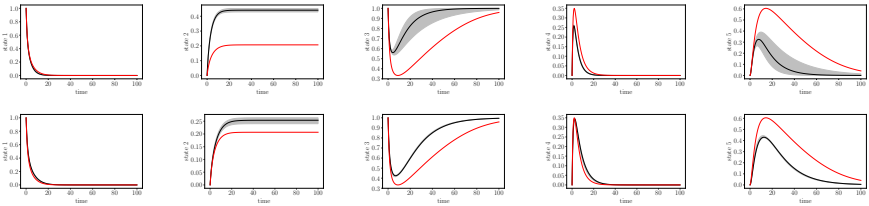


FIGURE A.2: Median plots for all states of the most difficult benchmark system in the literature, Protein Transduction. The red line is the ground truth, while the black line and the shaded area denote the median and the 75% quantiles of the results of 100 independent noise realizations. FGPGM (middle) is clearly able to find more accurate parameter estimates than AGM (top).

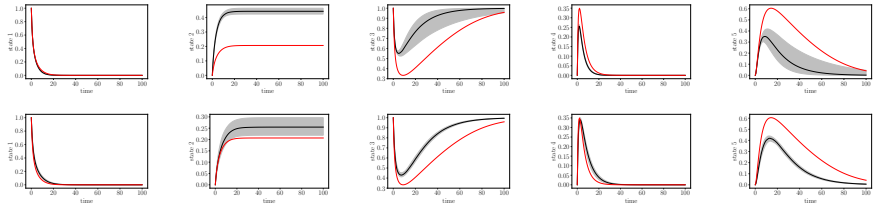


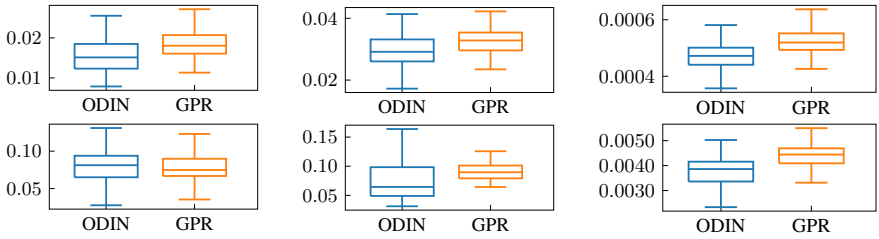
FIGURE A.3: Median Plots for all states of the most difficult benchmark system in the literature, Protein Transduction, for the high noise case.The red line is the ground truth, while the black line and the shaded area denote the median and the 75% quantiles of the results of 100 independent noise realizations. FGPGM (middle) is clearly able to find more accurate parameter estimates than AGM (top).

A.2 APPENDIX TO ODIN

A.2.1 *ODEs Provide Useful Information*

As shown in Figure A.4, ODIN provides more accurate state estimates in all but the high noise LV case compared to standard GP regression. The different behavior can be explained by the quality of the GP prior. For Lotka Volterra, the RBF kernel provides a perfectly suited prior for the sinusoidal form of its dynamics. It is thus not surprising that the GP regression estimates are already quite good, especially in a high noise setting. However, for both FitzHugh Nagumo and Protein Transduction, the GP prior is slightly off. Thus, including the additional information provided by the ODEs leads to significant improvements in state estimation.

A.2.2 *Median Trajectories*



(a) Lotka-Volterra (b) FitzHugh-Nagumo (c) Protein Transduction

FIGURE A.4: Comparing the RMSE of state estimates using vanilla GP regression and ODIN. All systems were evaluated on 100 independent noise realizations and parameter initializations. The top row shows the low noise case, the bottom row shows the high noise case.

A.2.3 *Kernel Approximation Error Bounds - Proofs*

In this section, we will extend classical quadrature results to prove the exponentially fast decaying error bounds for our kernel approximations. First, we will introduce some definitions we will use throughout this section. Then, we will be restating some useful lemmas and theorems. They will then be extended to a form which will finally allow us to prove the main theorem.

A.2.3.1 *Definitions*

First, let us start by defining some quantities similarly to the work of Mutny and Krause [MK18].

For any function f, let

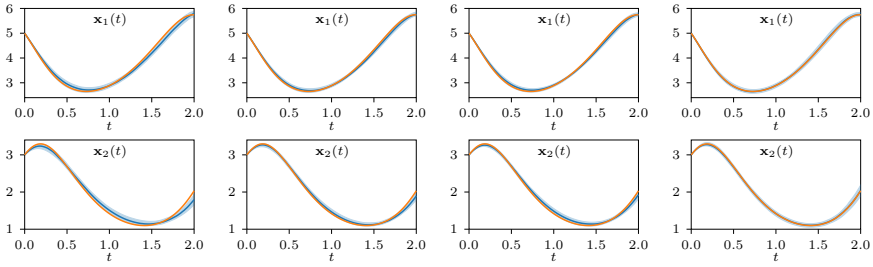
$$I(f(\omega)) := \int_{-\infty}^{+\infty} e^{-\omega^2} f(\omega) d\omega.$$

For any $r \in [0, 1]$, let

$$k(r) := \sqrt{\pi} e^{-\frac{r^2}{2l^2}} = \int_{-\infty}^{+\infty} e^{-\omega^2} cos\left(\omega r \frac{\sqrt{2}}{l}\right) d\omega = I\left(cos\left(\omega r \frac{\sqrt{2}}{l}\right)\right).$$

Let

$$Q_m(f) := \sum_{i=1}^{m} W_i^m f(x_i^m)$$



(a) AGM (b) FGPGM (c) ODIN (d) RKG3

FIGURE A.5: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the Lotka-Volterra system for $\sigma = 0.1$. The plot was created using 100 independent noise realizations, where the solid blue line is the median trajectory and the shaded areas denote the 25% and 75% quantiles. While the upper row shows the results for state x_1, the lower one does the same for state x_2. The orange trajectory is the ground truth. The difference between the four algorithms is barely visible.

denote the Gauss-Hermite quadrature scheme of order m and function f with weights $W_i^m \geq 0$ and abscissas x_i^m and let $S^m := \{x_1^m, x_2^m, ..., x_m^m\}$ denote the set of these abscissas.

Let $H_m(x)$ be the Hermite polynomial of order m and $h_m(x) := \frac{H_m(x)}{2^m}$ be its normalized version.

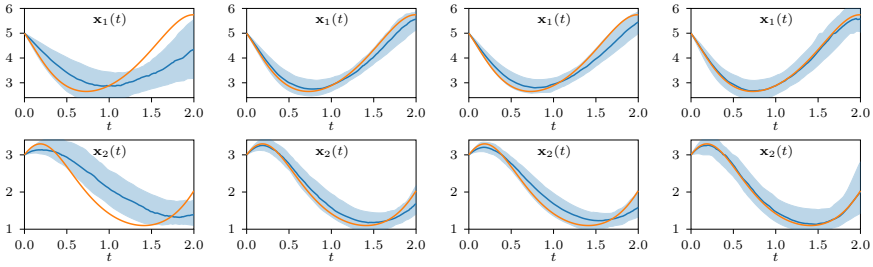
Let $E_m := \sqrt{\pi}\frac{1}{m^m}(\frac{e}{4l^2})^m$.

Using these definitions, we can restate the error bounds derived by Mutny and Krause [MK18] as

$$\left| I\left(\cos\left(\omega r\sqrt{2}/l\right)\right) - Q_m\left(\cos\left(\omega r\sqrt{2}/l\right)\right)\right| \leq E_m$$

and our bounds from Theorem 2 as

$$\begin{aligned}\frac{\sqrt{2}}{l}\left| I\left(\omega\sin\left(\omega r\frac{\sqrt{2}}{l}\right)\right) - Q_m\left(\omega\sin\left(\omega r\frac{\sqrt{2}}{l}\right)\right)\right| &\leq 8(m-1)E_{m-1} \\ &\leq \frac{2e}{l^2}E_{m-2}, \\ \frac{2}{l^2}\left| I\left(\omega^2\cos\left(\omega r\sqrt{2}/l\right)\right) - Q_m\left(\omega^2\cos\left(\omega r\sqrt{2}/l\right)\right)\right| &\leq \frac{4}{l^2}(m-1)E_{m-2} \\ &\leq \frac{2e}{l^4}E_{m-3}.\end{aligned}$$



(a) AGM (b) FGPGM (c) ODIN (d) RKG3

FIGURE A.6: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the Lotka-Volterra system for $\sigma = 0.5$. The plot was created using 100 independent noise realizations, where the solid blue line is the median trajectory and the shaded areas denote the 25% and 75% quantiles. While the upper row shows the results for state x_1, the lower one does the same for state x_2. The orange trajectory is the ground truth. While all algorithms seem to perform reasonably well, the perfect match between the sinusoidal dynamics and the RBF kernel lead to a well performing RKG3, while the more flexible Gaussian process based schemes seem to suffer more strongly from a smoothing bias.

A.2.3.2 *Useful Known Results*

Following Hildebrand [Hil87], we know that for all polynomials f with degree smaller equals $2m-1$, it holds that

$$Q_m(f(\omega)) = I(f(\omega))$$

and

$$I(h_m(\omega)^2) = \sqrt{\pi}\frac{m!}{2^m},$$

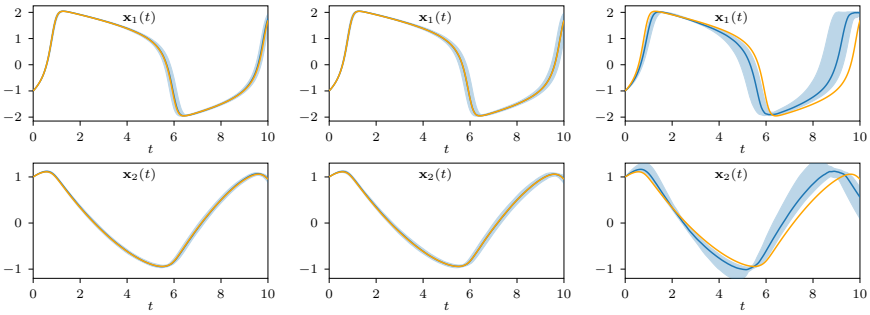
since

$$I(H_i(\omega)H_j(\omega)) = r_{ij}2^j j!\sqrt{\pi}.$$

In the following, we will need the two following, well-known Lemmas [see e.g. Hil87]:

Lemma 1 *Let f be a real function n times continuously differentiable and $q_n(x)$ a polynomial of degree $n-1$ that agrees with f at the set of distinct points $S = \{x_1, x_2, ..., x_n\}$. Let $\pi(x) = \prod_{i=1}^{n}(x - x_i)$. Then $\forall x \in \mathbb{R}$ we have*

$$f(x) - q_n(x) = \frac{f^{(n)}(\xi)}{n!}\pi(x)$$



(a) FGPGM (b) ODIN (c) RKG3

FIGURE A.7: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the FitzHugh-Nagumo system for a SNR of 100. The plot was created using 100 independent noise realizations, where the solid blue line is the median trajectory and the shaded areas denote the 25% and 75% quantiles. While the upper row shows the results for state x_1, the lower one does the same for state x_2. The orange trajectory indicates the ground truth. This experiment demonstrates how ODIN can comfortably deal with dynamics with rapidly changing lengthscales.

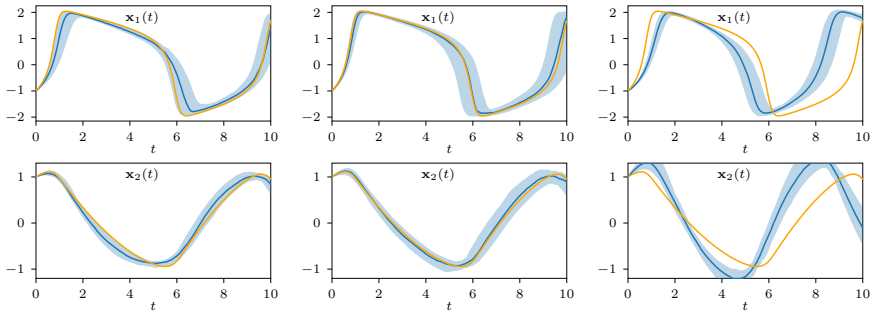
for some $\xi = \xi(x) \in I$, where I is the interval limited by the smallest and largest of the numbers $x_1, x_2, ..., x_n$ and x.

Lemma 2 *Let f be a real function that is $2n$-times continuously differentiable and $q_{2n}(x)$ a polynomial of degree $2n-1$ that agrees with f at the set of distinct points $S = \{x_1, x_2, ..., x_n\}$ and its derivative also agrees with f' at S. Let $\pi(x) = \prod_{i=1}^{n}(x - x_i)^2$. Then $\forall x \in \mathbb{R}$ we have*

$$f(x) - q_{2n}(x) = \frac{f^{(2n)}(\xi)}{(2n)!}\pi(x) \tag{A.1}$$

for some $\xi = \xi(x) \in I$, where I is the interval limited by the smallest and largest of the numbers $x_1, x_2, ..., x_n$ and x.

We will also need the main kernel approximation theorem of Mutny and Krause [MK18], namely



(a) FGPGM (b) ODIN (c) RKG3

FIGURE A.8: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the FitzHugh-Nagumo system for a SNR of 10. The plot was created using 100 independent noise realizations, where the solid blue line is the median trajectory and the shaded areas denote the 25% and 75% quantiles. While the upper row shows the results for state x_1, the lower one does the same for state x_2. The orange trajectory indicates the ground truth. This experiment demonstrates how ODIN can comfortably deal with dynamics with rapidly changing lengthscales.

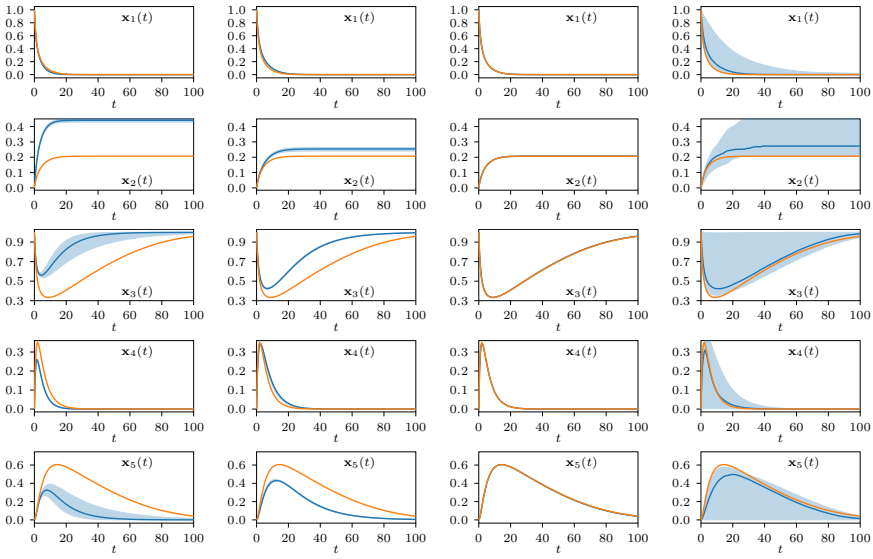
Theorem 5 *Consider the function* $cos(\frac{\sqrt{2}}{l}r\omega)$. *If we approximate the Gauss-Hermite quadrature scheme of order m the integral*

$$I\left(cos\left(\frac{\sqrt{2}}{l}r\omega\right)\right) = \int_{-\infty}^{+\infty} e^{-\omega^2} cos\left(\omega r\frac{\sqrt{2}}{l}\right) d\omega$$
$$\approx Q_m\left(cos\left(\frac{\sqrt{2}}{l}r\omega\right)\right),$$

we have

$$\left|I\left(cos\left(\frac{\sqrt{2}}{l}r\omega\right)\right) - Q_m\left(cos\left(\frac{\sqrt{2}}{l}r\omega\right)\right)\right| \leq E_m.$$

Without mentioning them further, we will always assume the following existence results: Let $S = \{x_1, x_2, \cdots x_n\}$ be a set of n distinct points in $\mathbb{R}$ and define $\pi(x) = \prod_{i=1}^{n}(x - x_i)$ and for $i = 1, ..., n$, $l_i(x) = \frac{\pi(x)}{(x-x_i)\pi'(x_i)}$. It holds that $l_i(x_j) = r_{ij}$ and that $deg(l_i) = n - 1$. Moreover, we define for $i = 1, ..., n$, $h_i(x) = l_i^2(x)(-2l_i'(x_i)(x - x_i) + 1)$ and $\bar{h}_i(x) = l_i^2(x)(x - x_i)$. It holds that $h_i(x_j) = r_{ij}$, $h_i'(x_j) = 0$, $\bar{h}_i(x_j) = 0$ and $\bar{h}_i'(x_j) = r_{ij}$ and



(a) AGM (b) FGPGM (c) ODIN (d) RKG3

FIGURE A.9: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the Protein Transduction system for $\sigma = 0.001$. The plot was created using 100 independent noise realizations, where the solid blue line is the median trajectory and the shaded areas denote the 23% and 75% quantiles. The orange trajectory is the ground truth. This experiment proves how ODIN is the only algorithm among the displayed ones that can handle non-Gaussian posterior marginals.

$deg(h_i) = deg(\bar{h}_i) = 2n - 1$ for $i = 1, ..., n$. Thus, we can directly state the following four Lemmas

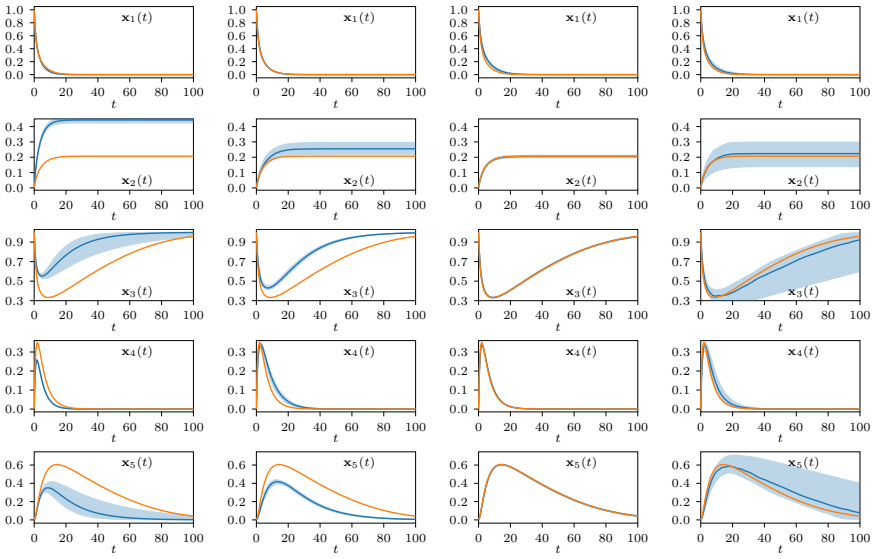
Lemma 3 *Let f be a real function and define*

$$y_1(x) := \sum_{i=1}^{n} f(x_i) l_i(x).$$

Then y_1 is a polynomial with degree $n - 1$ that agrees with f at S.

Lemma 4 *Let f be a real differential function and define*

$$y_2(x) := \sum_{i=1}^{n} f(x_i) h_i(x) + \sum_{i=1}^{n} f'(x_i) \bar{h}_i(x),$$



(a) AGM (b) FGPGM (c) ODIN (d) RKG3

FIGURE A.10: Comparison of the trajectories obtained by numerically integrating the inferred parameters of the Protein Transduction system for $\sigma = 0.01$. The plot was created using 100 independent noise realizations, where the solid blue line is the median trajectory and the shaded areas denote the 23% and 75% quantiles. The orange trajectory is the ground truth. As in the low noise case, this experiment proves how ODIN is the only algorithm among the displayed ones that can handle non-Gaussian posterior marginals.

Then y_2 is a polynomial with degree $2n-1$ that agrees with f at S and its derivative also agrees with the derivative in our proofs of f at S.

Lemma 5 *Consider a point $x_0 \notin S$. Let f be a real differential function and define*

$$y_3(x) := y_2(x) + \frac{f(x_0) - y_2(x_0)}{\pi^2(x_0)}\pi^2(x).$$

Then y_3 is a polynomial with degree $2n$ that agrees with f at $S \cup \{x_0\}$ and its derivative also agrees with the derivative of f at S.

Lemma 6 *Consider the point $x_1 \in S$. Let f be a real two times differential function and define $y_4(x) = y_2(x) + \frac{f''(x_1) - y_2''(x_1)}{2\pi'^2(x_1)}\pi^2(x)$. Then y_4 is a polynomial with*

degree $2n$ that agrees with f at S, its derivative also agrees with the derivative of f at S and its second derivative agrees with the second derivative of f at x_1.

A.2.3.3 *Intermediate Results*

For the proof of our theorem, we need to extend Lemmas 1 and 2. Here, we will introduce both the statements as well as the corresponding proofs. The proof technique is similar to the proofs of the original Lemmas 1 and 2 given by Hildebrand [Hil87].

Lemma 7 *Let f be a real, $2n+1$ times continuously differentiable function. Let $q_{2n+1}(x)$ be a polynomial of degree $2n$, which agrees with f at the set of distinct points $S = \{x_1, x_2, ..., x_n, x_*\}$ and whose derivatives agree with f' at $S \setminus \{x_*\}$. Let $\pi(x) = (x - x_*)\prod_{i=1}^{n}(x - x_i)^2$. Then $\forall x \in \mathbb{R}$, we have*

$$f(x) - q_{2n+1}(x) = \frac{f^{(2n+1)}(\xi)}{(2n+1)!}\pi(x)$$

for some $\xi = \xi(x) \in I$, where I is the interval limited by the smallest and largest of the numbers $x_1, x_2, ..., x_n, x_*$ and x.

Proof 2 *For $x \in S$ the property holds. Otherwise, fix $\bar{x} \in \mathbb{R}$ and define $F(x) := f(x) - q_{2n+1}(x) - K\pi(x)$, where K is chosen such that $F(\bar{x}) = 0$. As F has $n+2$ distinct roots at I, F' has $n+1$ distinct roots in I which are different from the points of S by Rolle's theorem (and F' vanishes at intermediate points). Similarly, since F' vanishes at all the points in $S \setminus \{x_*\}$, F' has at least $2n+1$ distinct roots in I. Thus, again by Rolle's theorem, F'' has $2n$ distinct roots in I. Continuing inductively, we see that $F^{(2n+1)}$ has one root $\xi \in I$. At ξ, we have*

$$0 = F^{(2n+1)}(\xi) = f^{(2n+1)}(\xi) - (2n+1)!K \Leftrightarrow K = \frac{f^{(2n+1)}(\xi)}{(2n+1)!}.$$

So

$$F(\bar{x}) = 0 = f(\bar{x}) - q_{2n+1}(\bar{x}) - \frac{f^{(2n+1)}(\xi)}{(2n+1)!}\pi(\bar{x})^2$$

$$\Leftrightarrow f(\bar{x}) - q_{2n+1}(\bar{x}) = \frac{f^{(2n+1)}(\xi)}{(2n+1)!}\pi(\bar{x})^2.$$

Lemma 8 *Let f be a real, $2n+1$ times continuously differentiable function. Let $q_{2n+1}(x)$ be a polynomial of degree $2n$, which agrees with f at the set of distinct points $S = \{x_1, x_2, ..., x_n\}$, whose derivatives agree with f' at S and*

whose second derivative at x_1 agrees with the second derivatives of f at x_1. Let $\pi(x) = (x - x_1)\prod_{i=1}^{n}(x - x_i)^2$. Then $\forall x \in \mathbb{R}$, we have

$$f(x) - q_{2n+1}(x) = \frac{f^{(2n+1)}(\xi)}{(2n+1)!}\pi(x)$$

for some $\xi = \xi(x) \in I$, where I is the interval limited by the smallest and largest of the numbers $x_1, x_2, ..., x_n$ and x.

Proof 3 *If $x \in S$, the property clearly holds. Otherwise, fix $\bar{x} \in \mathbb{R}$. Define $F(x) := f(x) - q_{2n+1}(x) - K\pi(x)$, where K is chosen such that $F(\bar{x}) = 0$. As F has $n+1$ distinct roots at I, F' has n distinct roots in I by Rolle's theorem. Note that these roots are different from the points of S, because F' vanishes at intermediate points). Since F' vanishes at all the points in S, F' has in total at least $2n$ distinct roots in I. Consequently, again by Rolle's theorem, F'' has $2n-1$ distinct roots in I. Note that these are all different from the points of S, because F'' vanishes at intermediate points. Since F'' vanishes at x_1, F'' vanishes at least at $2n$ points in total. Continuing inductively, we see that $F^{(2n+1)}$ has one root $\xi \in I$. At ξ, we have*

$$0 = F^{(2n+1)}(\xi) = f^{(2n+1)}(\xi) - (2n+1)!K \Leftrightarrow K = \frac{f^{(2n+1)}(\xi)}{(2n+1)!}.$$

So

$$F(\bar{x}) = 0 = f(\bar{x}) - q_{2n+1}(\bar{x}) - \frac{f^{(2n+1)}(\xi)}{(2n+1)!}\pi(\bar{x})^2$$
$$\Leftrightarrow f(\bar{x}) - q_{2n+1}(\bar{x}) = \frac{f^{(2n+1)}(\xi)}{(2n+1)!}\pi(\bar{x})^2.$$

Also, we will need the following two extensions of Theorem 5.

Lemma 9 *Using a Gauss-Hermite quadrature scheme of order m, we have*

$$\sqrt{2}\left|I(\omega\sin(\omega r\sqrt{2}/l)) - Q_m(\omega\sin(\omega r\sqrt{2}/l))\right| \leq 8l(m-1)E_{m-1}.$$

Proof 4 *For the proof of this theorem, we have to distinguish between two cases.*

CASE 1 *If m is even, $0 \in S^{m-1}$. By Lemma 5, there exists a polynomial y_{2m-3} of degree $2m-4$, which agrees with $\sin(\sqrt{2}r\omega/l)$ at S^{m-1} and whose derivates agree with the derivatives of $\sin(\sqrt{2}r\omega/l)$ at $S^{m-1} \setminus \{0\}$. By Lemma 7, we have for every ω that*

$$\omega\left(\sin(\sqrt{2}r\omega/l) - y_{2m-3}(\omega)\right) = (\sqrt{2}r/l)^{2m-3}\frac{\cos(\xi)}{(2m-3)!}h_{m-1}^2(\omega).$$

Using $r \leq 1$ and $|\cos(\xi)| \leq 1$, we get

$$\left|\omega\sin(\sqrt{2}r\omega/l) - \omega y_{2m-3}(\omega)\right| \leq (\sqrt{2}/l)^{2m-3}\frac{h_{m-1}^2(\omega)}{(2m-3)!}.$$

As h_{m-1}^2 has degree less than $2m$, we have $Q_m(h_{m-1}^2(\omega)) = I(h_{m-1}^2(\omega))$. Combining this with the fact that the weights W_i^m of a quadrature scheme are positive, it holds that

$$\begin{aligned}
&\left|Q_m(\omega y_{2m-3}(\omega)) - Q_m(\omega\sin(\omega r\sqrt{2}/l)\right| \\
&\qquad = \left|Q_m\left(\omega y_{2m-3}(\omega) - \omega\sin(\omega r\sqrt{2}/l)\right)\right| \\
&\qquad \leq Q_m\left(\left|\omega y_{2m-3}(\omega) - \omega\sin(\omega r\sqrt{2}/l)\right|\right) \\
&\qquad \leq Q_m\left((\sqrt{2}/l)^{2m-3}\frac{h_{m-1}^2(\omega)}{(2m-3)!}\right) \\
&\qquad = I\left((\sqrt{2}/l)^{2m-3}\frac{h_{m-1}^2(\omega)}{(2m-3)!}\right) \\
&\qquad = R_1,
\end{aligned}$$

where

$$R_1 := (\sqrt{2}/l)^{2m-3}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-3)!}.$$

Similarly,

$$\begin{aligned}
&\left|I(\omega y_{2m-3}(\omega)) - I(\omega\sin(\omega r\sqrt{2}/l)\right| \\
&= \left|I\left(\omega y_{2m-3}(\omega) - \omega\sin(\omega r\sqrt{2}/l)\right)\right| \\
&\leq I\left(\left|\omega y_{2m-3}(\omega) - \omega\sin(\omega r\sqrt{2}/l)\right|\right) \\
&\leq I\left((\sqrt{2}/l)^{2m-3}\frac{h_{m-1}^2(\omega)}{(2m-3)!}\right) \\
&= R_1.
\end{aligned}$$

As $\omega y_{2m-3}(\omega)$ has degree less than $2m$, we know that $Q_m\left(\omega y_{2m-3}(\omega)\right) = I\left(\omega y_{2m-3}(\omega)\right)$. This we can use to finally obtain

$$\begin{aligned}
&\frac{\sqrt{2}}{l}\left|I\left(\omega\sin(\omega r\sqrt{2}/l)\right) - Q_m\left(\omega\sin(\omega r\sqrt{2}/l)\right)\right| \\
&\quad\le \frac{\sqrt{2}}{l}\left|I\left(\omega\sin(\omega r\sqrt{2}/l)\right) - I\left(\omega y_{2m-3}(\omega)\right)\right| \\
&\qquad\quad + \frac{\sqrt{2}}{l}\left|Q_m\left(\omega y_{2m-3}(\omega)\right) - Q_m\left(\omega\sin(\omega r\sqrt{2}/l)\right)\right| \\
&\quad\le \sqrt{8}R_1/l \\
&\quad= 2(\sqrt{2}/l)^{2m-2}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-3)!} \\
&\quad\le 4(m-1)E_{m-1}.
\end{aligned}$$

CASE 2 *If m is odd, $0 \in S^{m-2}$. By Lemma 6, there exists a polynomial $y_{2m-3}(\omega)$ of degree $2m-4$, which agrees with $\sin(\sqrt{2}r\omega/l)$ at S^{m-2}, whose derivatives agree with the derivative of $\sin(\sqrt{2}r\omega)/l$ at S^{m-2}, and whose second derivative agrees with the second derivative of $\sin(\frac{\sqrt{2}}{l}r\omega)$ at 0. By Lemma 8, we have for every ω that*

$$\omega\left(\sin(\sqrt{2}r\omega/l) - y_{2m-3}(\omega)\right) = (2r/l)^{2m-3}\frac{\cos(\xi)}{(2m-3)!}\omega^2 h_{m-2}^2(\omega)$$

Using $r \le 1$ and $|\cos(\xi)| \le 1$, we get

$$\left|\omega\sin(\sqrt{2}r\omega/l) - \omega y_{2m-3}(\omega)\right| \le (\sqrt{2}/l)^{2m-3}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-3)!}.$$

As $\omega^2 h_{m-2}^2$ has degree less than $2m$, we have

$$Q_m(\omega^2 h_{m-2}^2(\omega)) = (\omega^2 h_{m-2}^2(\omega)).$$

Combining this with the fact that the weights W_i^m of a quadrature scheme are positive, it holds that

$$\begin{aligned}
&\left|Q_m(\omega y_{2m-3}(\omega)) - Q_m(\omega \sin(\sqrt{2}\omega r/l)\right| \\
&\qquad = \left|Q_m\left(\omega y_{2m-3}(\omega) - \omega \sin(\sqrt{2}\omega r/l)\right)\right| \\
&\qquad \leq Q_m\left(\left|\omega y_{2m-3}(\omega) - \omega \sin(\sqrt{2}\omega r/l)\right|\right) \\
&\qquad \leq Q_m\left((\sqrt{2}/l)^{2m-3}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-3)!}\right) \\
&\qquad = I\left((\sqrt{2}/l)^{2m-3}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-3)!}\right) \\
&\qquad = R_2,
\end{aligned}$$

where

$$R_2 = (\sqrt{2}/l)^{2m-3}\frac{I(\omega^2 h_{m-2}^2(\omega))}{(2m-3)!}.$$

Using the identity $xh_n(x) = h_{n+1}(x) + \frac{n}{2}h_{n-1}(x)$ the fact that

$$I(h_{n+1}(\omega)h_{n-1}(\omega)) = 0,$$

we obtain

$$\begin{aligned}
I\left(\omega^2 h_{m-2}^2(\omega)\right) &= I\left(h_{m-1}^2(\omega)\right) + ((m-2)/2)^2 I\left(h_{m-3}^2(\omega)\right) \\
&\leq 2\frac{(m-1)!\sqrt{\pi}}{2^{m-1}}.
\end{aligned}$$

Thus, R_2 can be bounded by

$$R_2 \leq 2(\sqrt{2}/l)^{2m-3}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-3)!}.$$

Similarly,

$$\begin{aligned}
&\left|I\left(\omega y_{2m-3}(\omega)\right) - I\left(\omega \sin(\sqrt{2}\omega r/l\right)\right| \\
&\qquad = \left|I\left(\omega y_{2m-3}(\omega) - \omega \sin\sqrt{2}(\omega r/l)\right)\right| \\
&\qquad \leq I\left(\left|\omega y_{2m-3}(\omega) - \omega \sin(\sqrt{2}\omega r/l)\right|\right) \\
&\qquad \leq I\left((\sqrt{2}/l)^{2m-3}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-3)!}\right) \\
&\qquad = R_2.
\end{aligned}$$

As $\omega y_{2m-3}(\omega)$ has degree less than $2m$, we know that $Q_m(\omega y_{2m-3}(\omega)) = I(\omega y_{2m-3}(\omega))$. This we can use to finally obtain

$$\begin{aligned}
&\frac{\sqrt{2}}{l}\left|I\left(\omega\sin(\sqrt{2}\omega r/l)\right) - Q_m\left(\omega\sin(\sqrt{2}\omega r/l)\right)\right| \\
&\quad\leq \frac{\sqrt{2}}{l}\left|I\left(\omega\sin(\sqrt{2}\omega r/l)\right) - I\left(\omega y_{2m-3}(\omega)\right)\right| \\
&\qquad + \frac{\sqrt{2}}{l}\left|Q_m\left(\omega y_{2m-3}(\omega)\right) - Q_m\left(\omega\sin(\sqrt{2}\omega r/l)\right)\right| \\
&\quad\leq \sqrt{8}R_2/l \\
&\quad= 4(\sqrt{2}/l)^{2m-2}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-3)!} \\
&\quad\leq 8(m-1)E_{m-1}.
\end{aligned}$$

Lemma 10 *Using a Gauss-Hermite quadrature scheme of order m, we have*

$$|I(\omega^2\cos(\sqrt{2}\omega r/l) - Q_m(\omega^2\cos(\sqrt{2}\omega r/l)| \leq 2(m-1)E_{m-2}.$$

Proof 5 *For the proof of this theorem, we have to distinguish between two cases.*

CASE 1 *If m is even, we know that $0 \in S^{m-1}$. By Lemma 5, there exists a polynomial $y_{2m-4}(\omega)$ of degree $2m-5$, which agrees with $\cos(\sqrt{2}\omega r/l)$ at $S^{m-1}\setminus\{0\}$ and whose derivatives agree with the derivatives of $\cos(\sqrt{2}\omega r/l)$ at $S^{m-1}\setminus\{0\}$. By Lemma 7, we have for every ω that*

$$\omega^2\left(\cos(\sqrt{2}r\omega/l) - y_{2m-4}(\omega)\right) = (\sqrt{2}r/l)^{2m-4}\frac{\cos(\xi)}{(2m-4)!}h_{m-1}^2(\omega).$$

Using $r \leq 1$ and $|cos(\xi)| \leq 1$, we get

$$\left|\omega^2\cos(\sqrt{2}r\omega/l) - \omega^2 y_{2m-4}(\omega)\right| \leq (\sqrt{2}/l)^{2m-4}\frac{h_{m-1}^2(\omega)}{(2m-4)!}.$$

As h_{m-1}^2 has degree less than $2m$, we have $Q_m(h_{m-1}^2(\omega)) = I(h_{m-1}^2(\omega))$. Combining this with the fact that the weights W_i^m of a quadrature scheme are positive, it holds that

$$\begin{aligned}
&\left|Q_m(\omega^2 y_{2m-4}(\omega)) - Q_m(\omega^2\cos(\sqrt{2}\omega r/l)\right| \\
&\qquad = \left|Q_m(\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l))\right| \\
&\qquad \le Q_m\left(\left|\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right|\right) \\
&\qquad \le Q_m\left((\sqrt{2}/l)^{2m-4}\frac{h_{m-1}^2(\omega)}{(2m-4)!}\right) \\
&\qquad = I\left((\sqrt{2}/l)^{2m-4}\frac{h_{m-1}^2(\omega)}{(2m-4)!}\right) \\
&\qquad = R_3
\end{aligned}$$

where

$$R_3 := (\sqrt{2}/l)^{2m-4}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-4)!}.$$

Similarly,

$$\begin{aligned}
&\left|I(\omega^2 y_{2m-4}(\omega)) - I(\omega^2\cos(\sqrt{2}\omega r/l))\right| \\
&\qquad = \left|I\left(\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right)\right| \\
&\qquad \le I\left(\left|\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right|\right) \\
&\qquad \le I((\sqrt{2}/l)^{2m-4}\frac{h_{m-1}^2(\omega)}{(2m-4)!}) \\
&\qquad = R_3.
\end{aligned}$$

As $\omega^2 y_{2m-4}(\omega)$ has degree less than $2m$, we know that $Q_m(\omega^2 y_{2m-4}(\omega)) = I(\omega^2 y_{2m-4}(\omega))$. Following a similar procedure as in the last step of the proof of Lemma 9, we get

$$\begin{aligned}
&\frac{2}{l^2}\left|I\left(\omega^2\cos(\sqrt{2}\omega r/l\right) - Q_m\left(\omega^2\cos(\sqrt{2}\omega r/l\right)\right| \\
&\qquad \le 4R_3/l^2 \\
&\qquad = 2(\sqrt{2}/l)^{2m-2}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-4)!} \\
&\qquad \le \frac{2}{l^2}(m-1)E_{m-2}.
\end{aligned}$$

CASE 2 *Suppose m is odd. By Lemma 4, there exists a polynomial $y_{2m-4}(\omega)$ of degree $2m-5$, which agrees with $\cos(\sqrt{2}r\omega/l)$ at S^{m-2} and whose derivatives agree with the derivatives of $\cos(\sqrt{2}r\omega/l)$ at S^{m-2}. By Lemma 2, we have for every ω that*

$$\omega^2(\cos(\sqrt{2}r\omega/l) - y_{2m-4}(\omega)) = (\sqrt{2}r/l)^{2m-4}\frac{\cos(\xi)}{(2m-4)!}\omega^2 h_{m-2}^2(\omega).$$

Using $r \leq 1$ and $|\cos(\xi)| \leq 1$, we get

$$\left|\omega^2\cos(\sqrt{2}r\omega/l) - \omega^2 y_{2m-4}(\omega)\right| \leq (\sqrt{2}/l)^{2m-4}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-4)!}.$$

As $\omega^2 h_{m-2}^2$ has degree less than $2m$, we have

$$Q_m(\omega^2 h_{m-2}^2(\omega)) = I(\omega^2 h_{m-2}^2(\omega)).$$

Combining this with the fact that the weights W_i^m of a quadrature scheme are positive, it holds that

$$\begin{aligned}
&\left|Q_m(\omega^2 y_{2m-4}(\omega)) - Q_m(\omega^2\cos(\sqrt{2}\omega r/l)\right| \\
&\qquad = \left|Q_m\left(\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right)\right| \\
&\qquad \leq Q_m\left(\left|\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right|\right) \\
&\qquad \leq Q_m\left((\sqrt{2}/l)^{2m-4}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-4)!}\right) \\
&\qquad = I\left((\sqrt{2}/l)^{2m-4}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-4)!}\right) \\
&\qquad = R_4
\end{aligned}$$

where

$$R_4 := (\sqrt{2}/l)^{2m-4}\frac{I(\omega^2 h_{m-2}^2(\omega))}{(2m-4)!}.$$

Using the identity $xh_n(x) = h_{n+1}(x) + \frac{n}{2}h_{n-1}(x)$ and the fact that

$$I(h_{n+1}(\omega)h_{n-1}(\omega)) = 0,$$

we obtain

$$I\left(\omega^2 h_{m-2}^2(\omega)\right) = I\left(h_{m-1}^2(\omega)\right) + (m-2/2)^2 I\left(h_{m-3}^2(\omega)\right)$$
$$\leq 2\frac{(m-1)!\sqrt{\pi}}{2^{m-1}}.$$

Thus, R_4 can be bounded by

$$R_4 \leq 2(\sqrt{2}/l)^{2m-4}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-4)!}.$$

Similarly,

$$\begin{aligned}
&\left|I\left(\omega^2 y_{2m-4}(\omega)\right) - I\left(\omega^2\cos(\sqrt{2}\omega r/l)\right)\right| \\
&\quad = \left|I\left(\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right)\right| \\
&\quad \leq I\left(\left|\omega^2 y_{2m-4}(\omega) - \omega^2\cos(\sqrt{2}\omega r/l)\right|\right) \\
&\quad \leq I\left((\sqrt{2}/l)^{2m-4}\frac{\omega^2 h_{m-2}^2(\omega)}{(2m-4)!}\right) \\
&\quad = R_4.
\end{aligned}$$

As $\omega^2 y_{2m-4}(\omega)$ has degree less than $2m$, we know that $Q_m(\omega^2 y_{2m-4}(\omega)) = I(\omega^2 y_{2m-4}(\omega))$. Following a similar procedure as in the last step of the proof of Lemma 9, we get

$$\begin{aligned}
&\left|I\left(\omega^2\cos(\sqrt{2}\omega r/l)\right) - Q_m\left(\omega^2\cos(\sqrt{2}\omega r/l)\right)\right| \\
&\quad \leq 2R \\
&\quad = 2l^2(\sqrt{2}/l)^{2m-2}\frac{(m-1)!\sqrt{\pi}}{2^{m-1}(2m-4)!} \\
&\quad \leq 2(m-1)E_{m-2}.
\end{aligned}$$

A.2.3.4 *Proof of main Theorem*

Given the previous results, the proof of Theorem 2 is now straight forward. Applying Lemma 9 to the feature map defined in Equation (3.26) provides the error bound on the kernel for the first derivative. Similarly, a combination of Lemma 10 and Equation (3.27) yields the error bound on the kernel for the second derivatives, finishing the proof of Theorem 2.

A.2.4 *Kernel Approximation Additional Plots*

A.2.5 *GP Regression with Derivatives*

In this section, we will further define and discuss our theoretical results on GP regression with derivatives. We will start by introducing all relevant definitions. Then, we will proof Theorem 3

A.2.5.1 *Problem Setting*

Consider the problem of Gaussian Process regression, using zero mean prior and the RBF kernel function $k_{\phi}(x,y) := \rho e^{-\frac{(x-y)^2}{2l^2}}$ for some fixed hyperparameters $\phi = (\rho, l)$, which denote the variance and the lengthscale. Suppose we are given at N observation points $\boldsymbol{t} := \boldsymbol{t}_N = (t_0, \ldots, t_{N-1})$ the N-dimensional (column) vectors $\boldsymbol{y} := \boldsymbol{x} + \boldsymbol{\epsilon}_{\sigma^2}$ and $\boldsymbol{F} := \dot{\boldsymbol{x}} + \boldsymbol{\epsilon}_{\gamma}$ of noisy state and noisy state derivative observations. Our goal is to obtain estimates of the values of state $x := x(T)$ and state derivative $\dot{x} := \dot{x}(T)$ at a new observation point T.

A.2.5.2 *Notation*

Let $\boldsymbol{k}_{\phi}(\boldsymbol{t}, T)$ denote the N dimensional kernel (column) vector, i.e.

$$\boldsymbol{k}_{\phi}(\boldsymbol{t}, T)_i := k_{\phi}(t_i, T). \tag{A.2}$$

Let ${}'k_{\phi}(x,y)$ denote the partial derivative of k_{ϕ} w.r.t. its first argument, i.e.

$${}'k_{\phi}(x,y) := \frac{\partial}{\partial a} k_{\phi}(a,b)|_{a=x,b=y}. \tag{A.3}$$

Let ${}'\boldsymbol{k}_{\phi}(\boldsymbol{t}, T)$ denote the N dimensional kernel derivative (column) vector, i.e.

$${}'\boldsymbol{k}_{\phi}(\boldsymbol{t}, T)_i := {}'k_{\phi}(t_i, T). \tag{A.4}$$

Let $k'_{\phi}(x,y)$ denote the partial derivative of k_{ϕ} w.r.t. its second argument, i.e.

$$k'_{\phi}(x,y) := \frac{\partial}{\partial b} k_{\phi}(a,b)|_{a=x,b=y}. \tag{A.5}$$

Let $\boldsymbol{k}'_{\phi}(\boldsymbol{t}, T)$ denote the N dimensional kernel derivative (column) vector, i.e.

$$\boldsymbol{k}'_{\phi}(\boldsymbol{t}, T)_i := k'_{\phi}(t_i, T). \tag{A.6}$$

Let $k''_{\phi}(x,y)$ denote the mixed partial derivative of k_{ϕ}, i.e.

$$k''_{\phi}(x,y) := \frac{\partial^2}{\partial a \partial b} k_{\phi}(a,b)|_{a=x,b=y}. \tag{A.7}$$

Let $\boldsymbol{k}''_{\phi}(\boldsymbol{t}, T)$ denote the N dimensional kernel derivative (column) vector, i.e.

$$\boldsymbol{k}''_{\phi}(\boldsymbol{t}, T)_i := k''_{\phi}(t_i, T). \tag{A.8}$$

Let $\boldsymbol{C}_{\phi}$ denote the $N \times N$ covariance kernel matrix, whose elements are given by

$$\left[\boldsymbol{C}_{\phi}\right]_{i,j} := k_{\phi}(t_i, t_j). \tag{A.9}$$

Let ${}'\boldsymbol{C}_{\phi}$ denote the kernel derivative matrix, whose elements are given by

$$\left[{}'\boldsymbol{C}_{\phi}\right]_{i,j} := \frac{\partial}{\partial a} k_{\phi}(a, b)|_{a=t_i, b=t_j}. \tag{A.10}$$

Let $\boldsymbol{C}'_{\phi}$ denote the kernel derivative matrix, whose elements are given by

$$\left[\boldsymbol{C}'_{\phi}\right]_{i,j} := \frac{\partial}{\partial b} k_{\phi}(a, b)|_{a=t_i, b=t_j}. \tag{A.11}$$

Let $\boldsymbol{C}''_{\phi}$ denote the mixed kernel derivative matrix, whose elements are given by

$$\left[\boldsymbol{C}''_{\phi}\right]_{i,j} := \frac{\partial^2}{\partial a \partial b} k_{\phi}(a, b)|_{a=t_i, b=t_j}. \tag{A.12}$$

In the graphical model of Figure 3.2, these matrices have been used to build the matrices

$$\boldsymbol{D} := \boldsymbol{C}'_{\phi} \boldsymbol{C}''^{-1}_{\phi} \tag{A.13}$$

and

$$\boldsymbol{A} := \boldsymbol{C}_{\phi} - \boldsymbol{C}'_{\phi} \boldsymbol{C}''^{-1}_{\phi} {}'\boldsymbol{C}_{\phi}. \tag{A.14}$$

Let $\hat{\boldsymbol{K}}_{\phi}$ denote the sum of the $2N \times 2N$ block matrix with the covariance matrix and its derivatives plus the diagonal noise matrix, i.e.

$$\hat{\boldsymbol{K}}_{\phi} := \begin{pmatrix} \boldsymbol{C}_{\phi} & \boldsymbol{C}'_{\phi} \\ {}'\boldsymbol{C}_{\phi} & \boldsymbol{C}''_{\phi} \end{pmatrix} + \begin{pmatrix} \sigma^2 \mathbb{I}_n & \boldsymbol{0} \\ \boldsymbol{0} & \gamma \mathbb{I}_n \end{pmatrix}. \tag{A.15}$$

Let $\hat{\boldsymbol{k}}_{\phi}(\boldsymbol{t}, T)$ denote the $2N$ dimensional (column) vector, which is a concatenation of $\boldsymbol{k}_{\phi}(\boldsymbol{t}, T)$ and ${}'\boldsymbol{k}_{\phi}(\boldsymbol{t}, T)$, i.e.

$$\hat{\boldsymbol{k}}_{\phi}(\boldsymbol{t}, T) := \begin{pmatrix} \boldsymbol{k}_{\phi}(\boldsymbol{t}, T) \\ {}'\boldsymbol{k}_{\phi}(\boldsymbol{t}, T) \end{pmatrix}. \tag{A.16}$$

Let $\hat{\boldsymbol{k}}'_{\phi}(\boldsymbol{t}, T)$ denote the $2N$ dimensional (column) vector, which is a concatenation of $\boldsymbol{k}'_{\phi}(\boldsymbol{t}, T)$ and $\boldsymbol{k}''_{\phi}(\boldsymbol{t}, T)$, i.e.

$$\hat{\boldsymbol{k}}'_{\phi}(\boldsymbol{t}, T) := \begin{pmatrix} \boldsymbol{k}'_{\phi}(\boldsymbol{t}, T) \\ \boldsymbol{k}''_{\phi}(\boldsymbol{t}, T) \end{pmatrix}. \tag{A.17}$$

Finally, we are able to write down the formulas for the scalar predictive mean and covariance at a new point T. Here, we let μ denote the mean of the state, μ' denote the mean of the derivative, α denote the variance of the state and α' the variance of the derivative prediction. They are given by

$$\mu(T) = \hat{\boldsymbol{k}}_{\phi}(\boldsymbol{t}, T)^T \hat{\boldsymbol{K}}_{\phi}^{-1} \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{F} \end{pmatrix}, \tag{A.18}$$

$$\mu'(T) = \hat{\boldsymbol{k}}'_{\phi}(\boldsymbol{t}, T)^T \hat{\boldsymbol{K}}_{\phi}^{-1} \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{F} \end{pmatrix}, \tag{A.19}$$

$$\alpha(T) = k(T, T) - \hat{\boldsymbol{k}}_{\phi}(\boldsymbol{t}, T)^T \hat{\boldsymbol{K}}_{\phi}^{-1} \hat{\boldsymbol{k}}_{\phi}(\boldsymbol{t}, T), \tag{A.20}$$

$$\alpha'(T) = k''(T, T) - \hat{\boldsymbol{k}}'_{\phi}(\boldsymbol{t}, T)^T \hat{\boldsymbol{K}}_{\phi}^{-1} \hat{\boldsymbol{k}}'_{\phi}(\boldsymbol{t}, T). \tag{A.21}$$

A.2.5.3 *Proof of Theorem 3*

In this Section, we will directly prove Theorem 3, the theorem regarding the approximation error of GPRD. Throughout this section, we will use $\tilde{v}$, $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{V}}$ to denote the feature approximation of any scalar v, vector $\boldsymbol{v}$ or matrix $\boldsymbol{V}$.

Proof 6 *Suppose that we use a QFF approximation scheme of order m to approximate the functions $k_{\phi}, {}'k_{\phi}, k'_{\phi}, k''_{\phi}$, which gives us a deterministic and uniform (over their domain) approximation guarantee of absolute error less than $\epsilon := \epsilon_{\phi}(m)$ (for any of them). W.l.o.g we can assume that the domain of these functions is $[0,1]^2$, $l \leq 1$ and $\rho \geq 1$ (so also $0 \leq T, t_1, \ldots t_n \leq 1$). Moreover, we assume that $|\boldsymbol{y}|_{max}, |\boldsymbol{F}|_{max} \leq R$, for some positive constant R.*

Using these assumptions, it is clear that

$$\left|\left|\hat{\boldsymbol{k}}_{\phi}(\boldsymbol{t}, T)\right|\right| \leq \sqrt{2N}\rho/l, \tag{A.22}$$

$$\left|\left|\hat{\boldsymbol{k}}'_{\phi}(\boldsymbol{t}, T)\right|\right| \leq \sqrt{2N}2\rho/l^2, \tag{A.23}$$

since $k_{\phi} \leq \rho$, ${}'k_{\phi} \leq \rho/l$ and $k''_{\phi} \leq 2\rho/l^2$.

Let $\boldsymbol{e}_1$ be the error (vector) when approximating $\hat{\boldsymbol{k}}_\phi(\boldsymbol{t}, T)$,

$$\boldsymbol{e}_1 \coloneqq \hat{\boldsymbol{k}}_\phi(\boldsymbol{t}, T) - \tilde{\hat{\boldsymbol{k}}}_\phi(\boldsymbol{t}, T), \tag{A.24}$$

whose norm, given our previous definitions, will be bounded by

$$||\boldsymbol{e}_1|| \leq \sqrt{2N}\epsilon. \tag{A.25}$$

Let $\boldsymbol{e}_2$ be the error (vector) when approximating $\hat{\boldsymbol{k}}_\phi(\boldsymbol{t}, T)$,

$$\boldsymbol{e}_2 \coloneqq \hat{\boldsymbol{k}}'_\phi(\boldsymbol{t}, T) - \tilde{\hat{\boldsymbol{k}}}'_{\boldsymbol{\phi}}(\boldsymbol{t}, T), \tag{A.26}$$

whose norm, given our previous definitions, will be bounded by

$$||\boldsymbol{e}_2|| \leq \sqrt{2N}\epsilon. \tag{A.27}$$

Let $\boldsymbol{E}_1$ be the error (matrix) when approximating $\hat{\boldsymbol{K}}_\phi$,

$$\boldsymbol{E}_1 \coloneqq \hat{\boldsymbol{K}}_{\boldsymbol{\phi}} - \tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}}, \tag{A.28}$$

whose spectral norm, given our previous definitions, will be bounded by

$$\sigma_1(\boldsymbol{E}_1) \leq 2N\epsilon. \tag{A.29}$$

Let $\boldsymbol{E}_2$be the error (matrix) when approximating $\hat{\boldsymbol{K}}_\phi^{-1}$,

$$\boldsymbol{E}_2 \coloneqq \hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1} - \tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}^{-1}}. \tag{A.30}$$

Since the matrix $\begin{pmatrix} \boldsymbol{C}_\phi & \boldsymbol{C}'_\phi \\ {}'\boldsymbol{C}_\phi & \boldsymbol{C}''_\phi \end{pmatrix}$ and its QFF approximation are symmetric and positive semi-definite, their smallest singular value is non-negative. Furthermore, $\begin{pmatrix} \sigma^2\mathbb{I}_N & \boldsymbol{0} \\ \boldsymbol{0} & \gamma\mathbb{I}_N \end{pmatrix}$ has smallest singular value c.

$$c \coloneqq \min\{\gamma, \sigma^2\}. \tag{A.31}$$

Thus, both $\hat{\boldsymbol{K}}$ and $\tilde{\hat{\boldsymbol{K}}}$ are symmetric, positive definite matrices, whose spectral norms are bounded by $\sigma_1(\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1}) \leq \frac{1}{c}$ and $\sigma_1(\tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}}^{-1}) \leq \frac{1}{c}$. Thus, using the matrix inversion Lemma, we can show that the spectral norm of $\boldsymbol{E}_2$ is bounded by

$$\sigma_1(\boldsymbol{E}_2) \leq \frac{2N}{c^2}\epsilon, \tag{A.32}$$

since $\boldsymbol{E}_2 = -\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1}\boldsymbol{E}_1\tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}^{-1}}$.

Combining all of these bounds, we get

$$|\epsilon_\mu| = \left|\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)^T\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1}\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{F}\end{pmatrix} - \left(\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)-\boldsymbol{e}_1\right)^T\left(\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1}-\boldsymbol{E}_1\right)\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{F}\end{pmatrix}\right| \tag{A.33}$$

$$= \left|\left(\boldsymbol{e}_1^T\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1} + \hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)^T\boldsymbol{E}_1 - \boldsymbol{e}_1^T\boldsymbol{E}_1\right)\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{F}\end{pmatrix}\right| \tag{A.34}$$

$$\leq \sqrt{2N}R\left(\frac{\sqrt{2N}}{c}\epsilon + \frac{\rho}{l}\sqrt{2N}2N\epsilon + \sqrt{2N}2N\epsilon^2\right) \tag{A.35}$$

$$\leq 10\frac{N^2R\rho}{lc}\epsilon, \tag{A.36}$$

where we used the fact that $\left|\left|\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{F}\end{pmatrix}\right|\right| \leq \sqrt{2N}R$.

Similarly, we get

$$|\epsilon_{\mu'}| = \left|\left(\boldsymbol{e}_2^T\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1} + \hat{\boldsymbol{k}}'_{\boldsymbol{\phi}}(\boldsymbol{t},T)^T\boldsymbol{E}_1 - \boldsymbol{e}_2^T\boldsymbol{E}_1\right)\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{F}\end{pmatrix}\right| \tag{A.37}$$

$$\leq \sqrt{2N}R\left(\frac{\sqrt{2N}}{c}\epsilon + \frac{\rho}{l^2}\sqrt{2N}4N\epsilon + \sqrt{2N}2N\epsilon^2\right) \tag{A.38}$$

$$\leq 14\frac{N^2R\rho}{l^2c}\epsilon. \tag{A.39}$$

Moreover,

$$|\epsilon_\Sigma| = |k(T,T) - \hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)^T\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1}\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T) - \tilde{k}(T,T) \tag{A.40}$$

$$+ (\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)-\boldsymbol{e}_1)^T(\hat{\boldsymbol{K}}_{\boldsymbol{\phi}}^{-1}-\boldsymbol{E}_1)(\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)-\boldsymbol{e}_1)| \tag{A.41}$$

$$\leq \epsilon + \left|-2\boldsymbol{e}_1^T\tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}^{-1}}\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T) + \boldsymbol{e}_1^T\tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}}^{-1}\boldsymbol{e}_1 - \hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)^T\boldsymbol{E}_2\hat{\boldsymbol{k}}_{\boldsymbol{\phi}}(\boldsymbol{t},T)\right| \tag{A.42}$$

$$\leq \epsilon + \frac{4N\rho}{lc}\epsilon + \frac{2N}{c}\epsilon^2 + \frac{4N^2\rho^2}{l^2c^2}\epsilon \tag{A.43}$$

$$\leq \frac{14N^2\rho^2}{l^2c^2}\epsilon \tag{A.44}$$

and

$$|\epsilon_{\Sigma'}| \leq \epsilon + \left| -2\boldsymbol{e}_2^T \tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}}^{-1} \hat{\boldsymbol{k}}'_{\boldsymbol{\phi}}(\boldsymbol{t}, T) + \boldsymbol{e}_2^T \tilde{\hat{\boldsymbol{K}}}_{\boldsymbol{\phi}}^{-1} \boldsymbol{e}_2 - \hat{\boldsymbol{k}}'_{\boldsymbol{\phi}}(\boldsymbol{t}, T)^T \boldsymbol{E}_2 \hat{\boldsymbol{k}}'_{\boldsymbol{\phi}}(\boldsymbol{t}, T) \right| \tag{A.45}$$

$$\leq \epsilon + \frac{8N\rho}{l^2 c}\epsilon + \frac{2N}{c}\epsilon^2 + \frac{16N^2\rho^2}{l^4 c^2}\epsilon \tag{A.46}$$

$$\leq \frac{27N^2\rho^2}{l^4 c^2}\epsilon. \tag{A.47}$$

Taking the maximum of the four bounds, we can observe that

$$|e_{tot}| := |\max\{e_{\tilde{\mu}}, e_{\tilde{\alpha}}, e_{\tilde{\mu}'}, e_{\tilde{\alpha}'}\}| \leq \frac{27N^2\rho^2 R}{l^4 c^2}\epsilon. \tag{A.48}$$

For any $0 < C < 1$*, we can choose* $M \in \mathbb{N}$ *such that*

$$M \geq \max\left\{ \frac{e}{2l^2}, \log\left(\frac{270N^2\rho^3 R}{l^8 c^2 C} \right) \right\}. \tag{A.49}$$

Then, choose the order of quadrature m *according to* $M := m - 3$*. With this choice, it holds that*

$$\left(\frac{e}{4l^2 M}\right)^M \leq \frac{l^8 c^2}{270N^2\rho^3 R} C \tag{A.50}$$

$$\implies \frac{2e\sqrt{\pi}\rho}{l^4}\left(\frac{e}{4l^2 M}\right)^M \leq \frac{l^4 c^2}{27N^2\rho^2 R} C \tag{A.51}$$

$$\implies \epsilon \leq \frac{l^4 c^2}{27N^2\rho^2 R} C \tag{A.52}$$

$$\iff \frac{27N^2\rho^2 R}{l^4 c^2}\epsilon \leq C, \tag{A.53}$$

$$\tag{A.54}$$

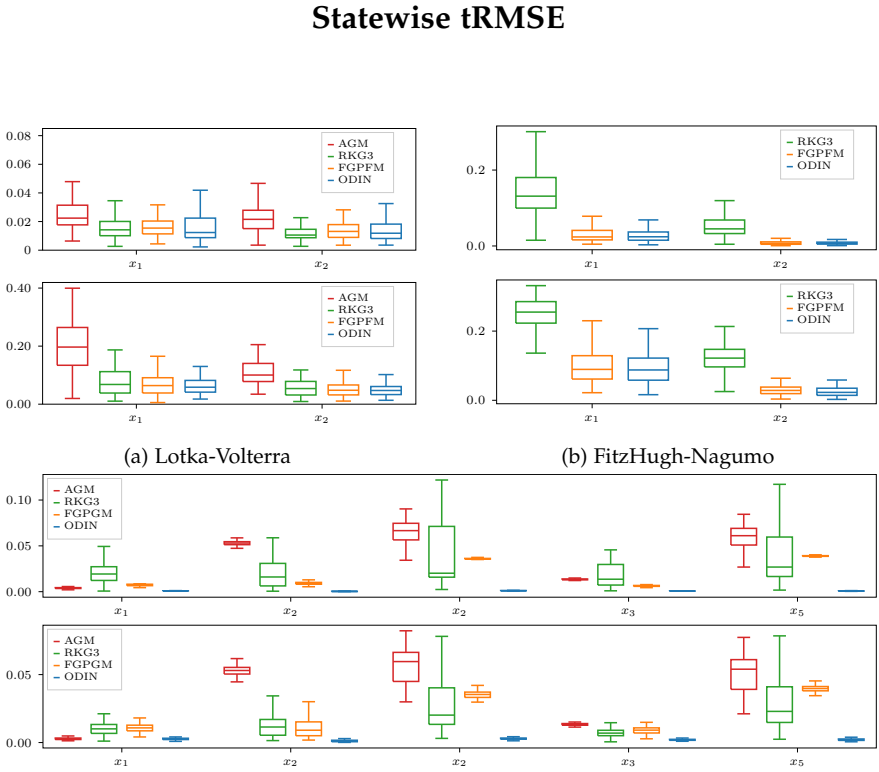
which concludes the proof of this theorem.

The above bound could be reformulated as $m \geq 12 + \max\left\{ \frac{e}{2l^2}, \log\left(\frac{N^2\rho^3 R}{l^8 c^2 C} \right) \right\}$. Moreover, we assumed that $R \geq 1$ and that $c \leq 1$. If any of these conditions is not met, then the bounds are still valid if we substitute these quantities by 1 (the same holds also for ρ and l). Finally, we implicitly assumed that ϵ, the uniform upper bound of the approximation for $k_{\boldsymbol{\phi}}, {}' k_{\boldsymbol{\phi}}, k'_{\boldsymbol{\phi}}, k''_{\boldsymbol{\phi}}$ is smaller than 1. As seen in Section A.2.3, this happens if $m \geq 3 + \max\left\{ \frac{e}{2l^2}, \log\left(\frac{10\rho}{l^4} \right) \right\}$, a condition which is met if $m \geq 3 + \max\left\{ \frac{e}{2l^2}, \log\left(\frac{270N^2\rho^3 R}{l^8 c^2 C} \right) \right\}$.

A.2.6 *Additional Empirical Evaluation GPR*

A.2.6.1 *Lotka Volterra*

A.2.6.2 *Protein Transduction*

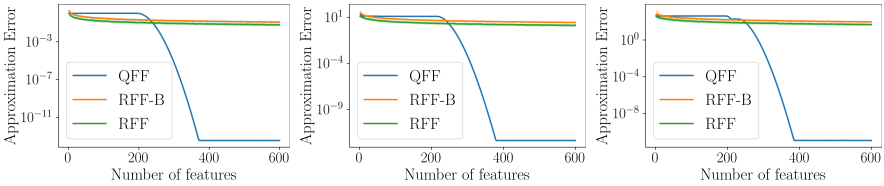


(a) Lotka-Volterra

(b) FitzHugh-Nagumo

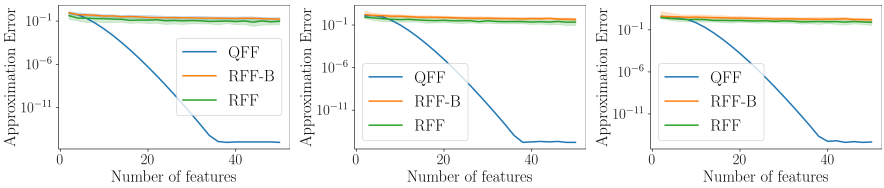
(c) Protein Transduction

FIGURE A.11: Statewise trajectory RMSE for all benchmark systems in the parameter inference problem. For each pair of plots, the top shows the low noise case with $\sigma = 0.1$ for LV, $\sigma = 0.001$ for PT and $SNR = 100$ for FHN. The bottom shows the high noise case with $\sigma = 0.5$ for LV, $\sigma = 0.01$ for PT and $SNR = 10$ for FHN.



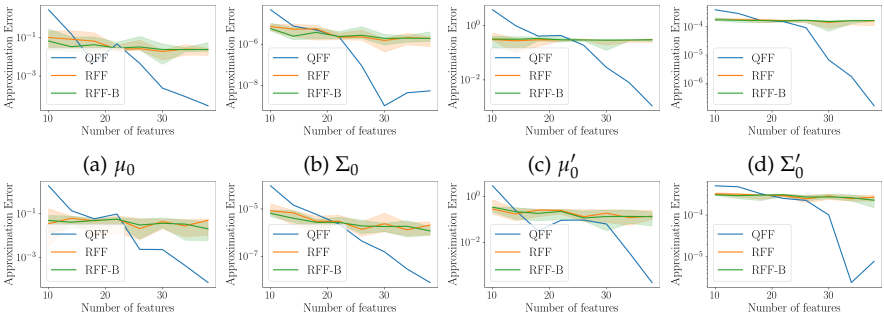
(a) $k(r)$ (b) $k'(r)$ (c) $k''(r)$

FIGURE A.12: Comparing the maximum error of different feature expansions over $r \in [0, 1]$. For the stochastic RFF and RFF-B, median, 12.5% and 87.5% quantiles over 100 random samples are shown, but barely visible due to the exponential decay of the error of the QFF error. As predicted by our theoretical analysis, the error is a bit higher for the derivatives, but still decaying exponentially. In this plot, we set $l = 0.05$.



(a) $k(r)$ (b) $k'(r)$ (c) $k''(r)$

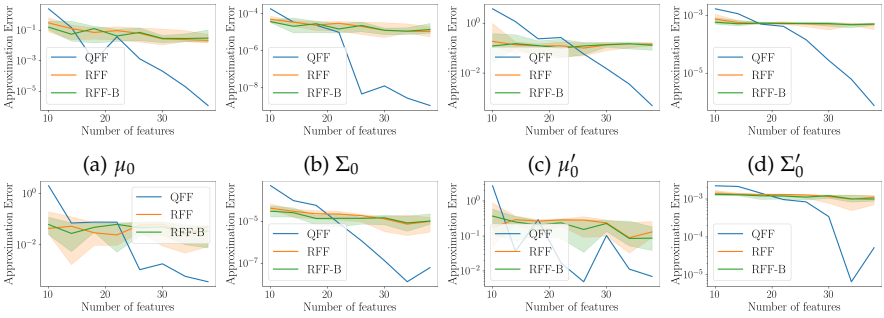
FIGURE A.13: Comparing the maximum error of different feature expansions over $r \in [0, 1]$. For the stochastic RFF and RFF-B, median, 12.5% and 87.5% quantiles over 100 random samples are shown, but barely visible due to the exponential decay of the error of the QFF error. As predicted by our theoretical analysis, the error is a bit higher for the derivatives, but still decaying exponentially. In this plot, we set $l = 0.5$.



(a) μ_0 (b) Σ_0 (c) μ_0' (d) Σ_0'

(e) μ_1 (f) Σ_1 (g) μ_1' (h) Σ_1'

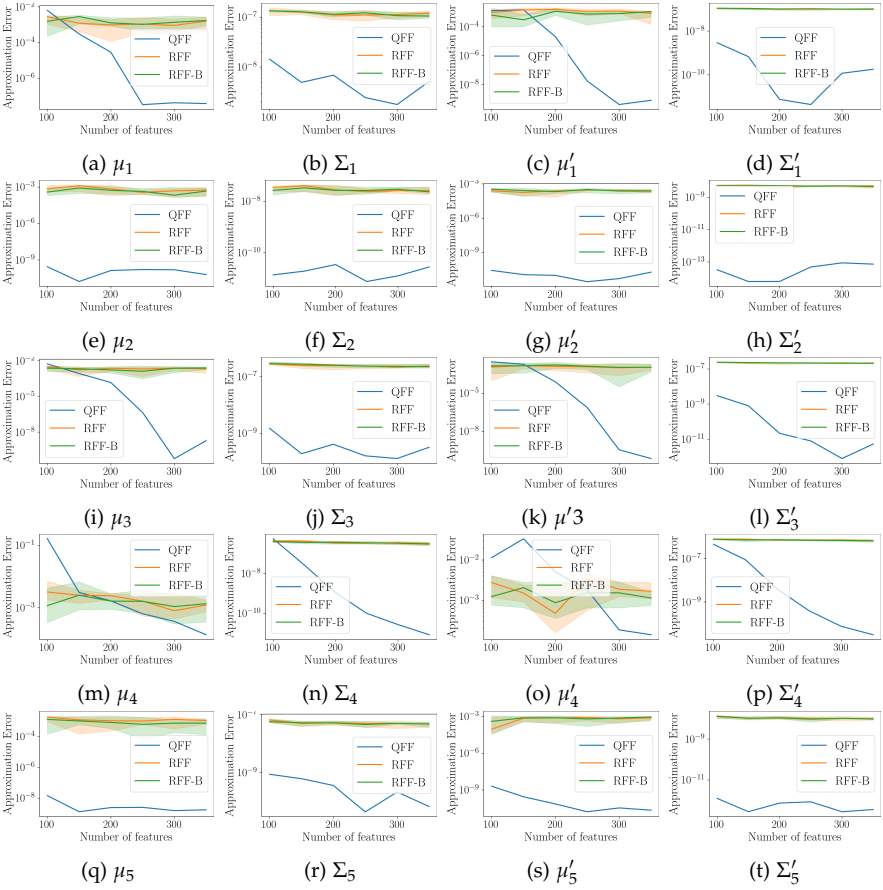
FIGURE A.14: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 1.75$ for the Lotka Volterra system with 1000 observations and $\sigma^2 = 0.1$. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.



(a) μ_0 (b) Σ_0 (c) μ_0' (d) Σ_0'

(e) μ_1 (f) Σ_1 (g) μ_1' (h) Σ_1'

FIGURE A.15: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 1.75$ for the Lotka Volterra system with 1000 observations and $\sigma^2 = 0.5$. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.



(a) μ_1 (b) Σ_1 (c) μ'_1 (d) Σ'_1

(e) μ_2 (f) Σ_2 (g) μ'_2 (h) Σ'_2

(i) μ_3 (j) Σ_3 (k) $\mu'3$ (l) Σ'_3

(m) μ_4 (n) Σ_4 (o) μ'_4 (p) Σ'_4

(q) μ_5 (r) Σ_5 (s) μ'_5 (t) Σ'_5

FIGURE A.16: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 30$ for the Protein Transduction system with 1000 observations and $\sigma^2 = 0.0001$. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.

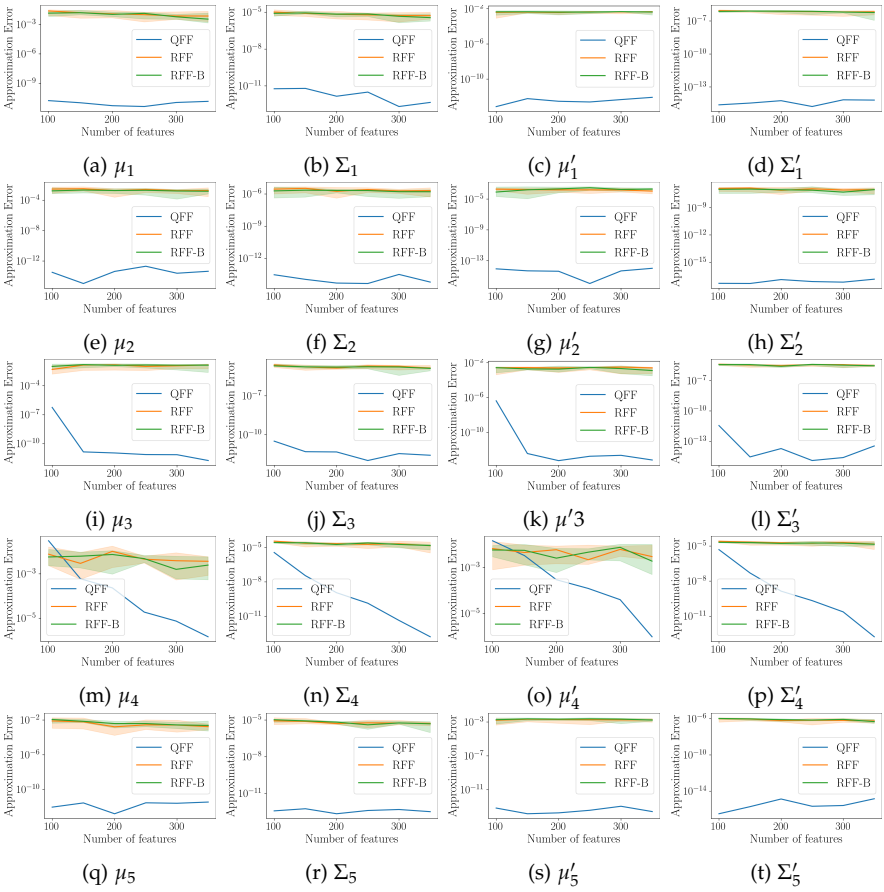
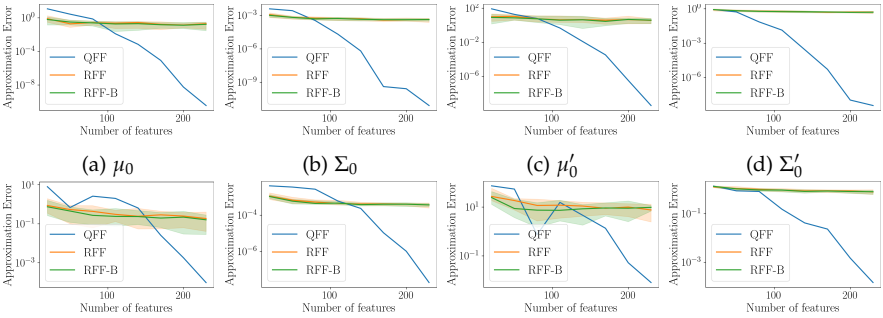


FIGURE A.17: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 30$ for the Protein Transduction system with 1000 observations and $\sigma^2 = 0.01$. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.

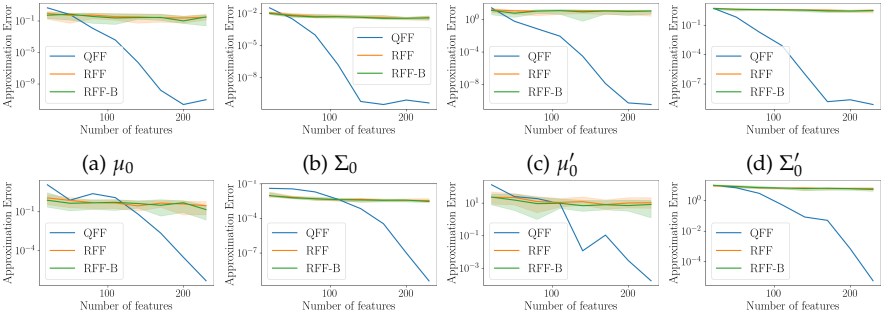
A.2.6.3 *Lorenz*



(a) μ_0 (b) Σ_0 (c) μ_0' (d) Σ_0'

(e) μ_1 (f) Σ_1 (g) μ_1' (h) Σ_1'

FIGURE A.18: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 0.8$ for the Lorenz system with 1000 observations and an SNR of 100. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.



(a) μ_0 (b) Σ_0 (c) μ_0' (d) Σ_0'

(e) μ_1 (f) Σ_1 (g) μ_1' (h) Σ_1'

FIGURE A.19: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 0.8$ for the Lorenz system with 1000 observations and an SNR of 10. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.

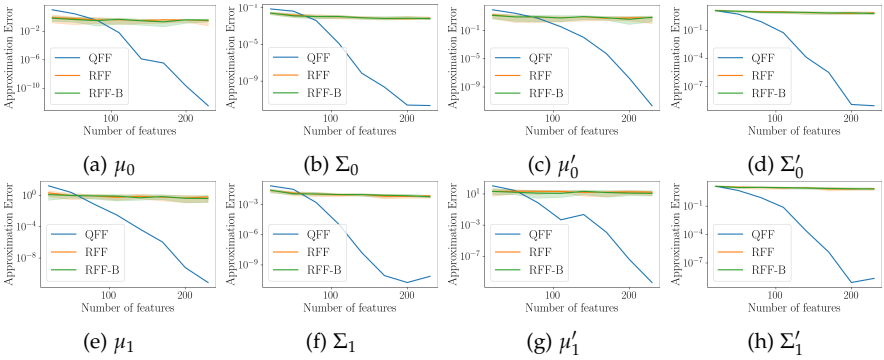


FIGURE A.20: Approximation error of the different feature approximations compared to the accurate GP, evaluated at $t = 0.8$ for the Lorenz system with 1000 observations and an SNR of 5. For each feature, we show the median as well as the 12.5% and 87.5% quantiles over 10 independent noise realizations, separately for each state dimension.

A.2.7 *Risk Approximation Error Bounds*

In this section, we will provide all necessary results to prove Theorem 4. We will start by introducing necessary definitions, after which we will formulate and prove lemmas for all terms involved in the risk. Finally, we will combine these lemmas to form the main proof of the theorem.

A.2.7.1 *Definitions*

Let

$$\mathcal{R}_{\lambda\gamma\phi}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y}) := \boldsymbol{x}^T(\boldsymbol{C}_\phi + \lambda \boldsymbol{I})^{-1}\boldsymbol{x} \tag{A.55}$$

$$+ (\boldsymbol{x} - \boldsymbol{y})^T\sigma^{-2}(\boldsymbol{x} - \boldsymbol{y}) \tag{A.56}$$

$$+ (\boldsymbol{f} - \boldsymbol{D}\boldsymbol{x})^T(\boldsymbol{A} + \gamma \boldsymbol{I})^{-1}(\boldsymbol{f} - \boldsymbol{D}\boldsymbol{x}) \tag{A.57}$$

be the original risk of ODIN, where all matrices are chosen as defined in Section A.2.5.

By writing $\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}$, $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{D}}$ for the QFF-approximated quantities as described in Section A.2.5, we get the approximate risk function

$$\tilde{\mathcal{R}}_{\lambda\gamma\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\theta}) := \boldsymbol{x}^T(\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{x} \tag{A.58}$$

$$+ (\boldsymbol{x}-\boldsymbol{y})^T\sigma^{-2}(\boldsymbol{x}-\boldsymbol{y}) \tag{A.59}$$

$$+ (\boldsymbol{f}-\tilde{\boldsymbol{D}}\boldsymbol{x})^T(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\tilde{\boldsymbol{D}}\boldsymbol{x}). \tag{A.60}$$

Let

$$|\boldsymbol{K}|_F := \sqrt{\mathrm{tr}(\boldsymbol{K}\boldsymbol{K}^T)} \tag{A.61}$$

be the Frobenius norm of a matrix $\boldsymbol{K}$. Furthermore, let

$$|\boldsymbol{K}|_{\max} := \max_{i,j} |\boldsymbol{K}_{ij}| \tag{A.62}$$

denote the max-norm of $\boldsymbol{K}$ and

$$\sigma_1(\boldsymbol{K}) := |\boldsymbol{K}|_2 \tag{A.63}$$

denote the spectral norm of $\boldsymbol{K}$, given by its largest singular value.

A.2.7.2 *Intermediate Results*

It is well-known that $|\boldsymbol{K}\boldsymbol{x}| \leq \sigma_1(\boldsymbol{K})|\boldsymbol{x}|$ and that, for a $N \times N$ matrix $\boldsymbol{K}$, we have

$$\sigma_1(\boldsymbol{K}) \leq \sqrt{\sum \sigma_i^2(\boldsymbol{K})} = \sqrt{|\boldsymbol{K}|_F^2} \leq \sqrt{N^2|\boldsymbol{K}|_{max}^2} = N|\boldsymbol{K}|_{max} \tag{A.64}$$

In particular, for an RBF kernel as we have chosen, it holds that

$$\sigma_1(\boldsymbol{C}_{\boldsymbol{\phi}}) \leq \rho N \tag{A.65}$$

and

$$\sigma_1(\boldsymbol{C}'_{\boldsymbol{\phi}}) \leq \rho N/l \tag{A.66}$$

since for all x, it holds that $xe^{-\frac{x^2}{2}} \leq \frac{1}{\sqrt{e}} \leq 1$.

Defining the error matrix $\boldsymbol{E}_1 := \boldsymbol{E}_1(m) = \tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} - \boldsymbol{C}_{\boldsymbol{\phi}}$, we know that $|\boldsymbol{E}_1|_{max} \leq \sqrt{\frac{\pi}{2}}\frac{1}{m^m}(\frac{e}{4l^2})^m$. Thus, for

$$a := \frac{e}{4l^2} \tag{A.67}$$

we have

$$\sigma_1(\boldsymbol{E}_1) \leq 2N\left(\frac{a}{m}\right)^m. \tag{A.68}$$

To avoid overly complicated notation, we shall assume that $\gamma \leq 1, \lambda \leq 1, \rho \geq 1$, $l \leq 1$ (and specifically $l \leq \frac{e}{4}$). If any of these assumptions is violated, the corresponding parameter could be substituted by 1 in all final bounds and our results would still be valid.

In our first lemma, we will bound the first term of the risk

Lemma 11 *For some jitter $\lambda > 0$, consider the term $\boldsymbol{x}^T(\boldsymbol{C_\phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x}$ approximated by $x^T(\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}x$. Then for $m \geq M := 7 + \max\left\{\frac{e}{2l^2}, \log\left(\frac{\rho^2 n^3}{\lambda^2 \gamma l^4}\right)\right\}$, it holds that*

$$\left|\boldsymbol{x}^T(\boldsymbol{C_\phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x} - \boldsymbol{x}^T(\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x}\right| \leq \epsilon_1 \boldsymbol{x}^T(\boldsymbol{C_\phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x} \quad \text{(A.69)}$$

where $\epsilon_1 = \frac{4N}{\lambda}(\frac{a}{m})^m$.

Proof 7 *Define $\boldsymbol{K} := \boldsymbol{C_\phi} + \lambda \boldsymbol{I}$. Since $\boldsymbol{C_\phi}$ is positive-definite and $\lambda > 0$, $\sigma_1(\boldsymbol{K}^{-1}) < \frac{1}{\lambda}$. Clearly, $m \geq \max\left\{\frac{e}{2l^2}, \log\left(\frac{4N}{\lambda}\right)\right\}$, as $m \geq M$. Thus, setting $\boldsymbol{E} := \boldsymbol{E}_1$, we have for any vector $\boldsymbol{u}$ with $||\boldsymbol{u}||_2 = 1$*

$$\left|\left|\left(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1}\right)\boldsymbol{u}\right|\right| \geq ||\boldsymbol{u}|| - \left|\left|\boldsymbol{E}\boldsymbol{K}^{-1}\boldsymbol{u}\right|\right| \geq 1 - \sigma_1(\boldsymbol{E})\sigma_1(\boldsymbol{K}^{-1}) \geq 1 - \frac{2n}{\lambda}(\frac{a}{m})^m \geq \frac{1}{2}. \quad \text{(A.70)}$$

Thus, $\sigma_1(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1}) \geq \frac{1}{2}$, which implies $\sigma_1((\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1})^{-1}) \leq 2$.

Since, $\sigma_1(\boldsymbol{K}^{-1}(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1})^{-1}\boldsymbol{E}\boldsymbol{K}^{-1}) \leq \sigma_1(\boldsymbol{K}^{-1})^2\sigma_1((\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1})^{-1})\sigma_1(\boldsymbol{E}) \leq \frac{4N}{\lambda^2}(\frac{a}{m})^m$, we have

$$\sigma_1\left((\boldsymbol{C_\phi} + \lambda \boldsymbol{I})^{-1} - (\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\right) \leq \frac{4N}{\lambda^2}\left(\frac{a}{m}\right)^m. \quad \text{(A.71)}$$

Combining these inequalities, we finally get

$$\begin{aligned}
&||\boldsymbol{x}^T(\boldsymbol{C_\phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x} - \boldsymbol{x}^T(\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x}|| && \text{(A.72)}\\
&= \left|\left|\boldsymbol{x}^T\boldsymbol{K}^{-1}(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1})^{-1}\boldsymbol{E}\boldsymbol{K}^{-1}\boldsymbol{x}\right|\right| && \text{(A.73)}\\
&\leq \left|\left|\boldsymbol{x}^T\boldsymbol{K}^{-\frac{1}{2}}\right|\right| \sigma_1\left(\boldsymbol{K}^{-\frac{1}{2}}(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1})^{-1}\boldsymbol{E}\boldsymbol{K}^{-\frac{1}{2}}\right) \left|\left|\boldsymbol{x}^T\boldsymbol{K}^{-\frac{1}{2}}\right|\right| && \text{(A.74)}\\
&\leq \left|\left|\boldsymbol{x}^T\boldsymbol{K}^{-\frac{1}{2}}\right|\right|^2 \sigma_1(\boldsymbol{K}^{-\frac{1}{2}})^2\sigma_1((\boldsymbol{I} + \boldsymbol{E}\boldsymbol{K}^{-1})^{-1})\sigma_1(\boldsymbol{E} && \text{(A.75)}\\
&\leq \frac{4N}{\lambda}\left(\frac{a}{m}\right)^m \left|\left|\boldsymbol{x}^T(\boldsymbol{C_\phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x}\right|\right|. && \text{(A.76)}
\end{aligned}$$

Before dealing with the third term, we introduce two lemmas providing an upper bound for the spectral norm of the two error matrices

$$\boldsymbol{E}_2 = \boldsymbol{E}_2(m) := \tilde{\boldsymbol{A}} - \boldsymbol{A} \quad \text{(A.77)}$$

and

$$\boldsymbol{E}_3 = \boldsymbol{E}_3(m) := \tilde{\boldsymbol{D}} - \boldsymbol{D}. \tag{A.78}$$

Lemma 12 *Let* $m \geq M := 7 + \max\left\{\frac{e}{2l^2}, \log\left(\frac{\rho^2 N^3}{\lambda^2 \gamma l^4}\right)\right\}$. *Then*

$$\sigma_1\left(\boldsymbol{E}_2(m+3)\right) \leq 20\frac{\rho^2 n^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m. \tag{A.79}$$

Proof 8 *We have*

$$\boldsymbol{E}_2 = \tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}'' - \boldsymbol{C}_{\boldsymbol{\phi}}'' + {}'\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}(\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}' - {}'\boldsymbol{C}_{\boldsymbol{\phi}}(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{C}_{\boldsymbol{\phi}}', \tag{A.80}$$

which we decompose by defining

$$\boldsymbol{E}_{21} := \tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}' - \boldsymbol{C}_{\boldsymbol{\phi}}', \tag{A.81}$$

$$\boldsymbol{E}_{22} := (\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1} - (\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}, \tag{A.82}$$

$$\boldsymbol{E}_{23} := \tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}'' - \boldsymbol{C}_{\boldsymbol{\phi}}''. \tag{A.83}$$

Since $||\boldsymbol{E}_{21}(m+2)||_{\max} \leq 16a\left(\frac{a}{m}\right)^m$, *it holds that*

$$\sigma_1(\boldsymbol{E}_{21}(m+3)) \leq 16aN\left(\frac{a}{m}\right)^m. \tag{A.84}$$

Moreover, since $||\boldsymbol{E}_{23}(m+3)||_{\max} \leq 32a^2\left(\frac{a}{m}\right)^m$, *it holds that*

$$\sigma_1(\boldsymbol{E}_{23}(m+3)) = 32a^2N\left(\frac{a}{m}\right)^m. \tag{A.85}$$

Finally, as stated by Equation (A.71),

$$\sigma_1(\boldsymbol{E}_{22}(m)) \leq \frac{4N}{\lambda^2}\left(\frac{a}{m}\right)^m. \tag{A.86}$$

Combining all of those, we get

$$\sigma_1(\boldsymbol{E}_2(m+3)) \tag{A.87}$$

$$= \sigma_1\left(\boldsymbol{E}_{23} + ({}'\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{E}_{21}^T)((\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1} + \boldsymbol{E}_{22})(\boldsymbol{C}_{\boldsymbol{\phi}}' + \boldsymbol{E}_{21}) - {}'\boldsymbol{C}_{\boldsymbol{\phi}}(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{C}_{\boldsymbol{\phi}}'\right) \tag{A.88}$$

$$\leq \sigma_1(\boldsymbol{E}_{23}) + 2\sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}})\sigma_1((\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1})\sigma_1(\boldsymbol{E}_{21}) + \sigma_1(\boldsymbol{E}_{21})^2\sigma_1((\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}) \tag{A.89}$$

$$+ \sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}})^2\sigma_1(\boldsymbol{E}_{22}) + 2\sigma_1(\boldsymbol{E}_{21})\sigma_1(\boldsymbol{E}_{22})\sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}}) + \sigma_1(\boldsymbol{E}_{21})^2\sigma_1(\boldsymbol{E}_{22}) \tag{A.90}$$

$$\leq 20\frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m \tag{A.91}$$

For the last inequality, we need the following facts: Since $m > M$, it holds that

$$\sigma_1(\boldsymbol{E}_{23}) \leq 32a^2 \left(\frac{a}{m}\right)^m N \leq \frac{1}{2}\frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.92}$$

$$2\sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}})\sigma_1((\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-1})\sigma_1(\boldsymbol{E}_{21}) \leq 32\frac{\rho a}{l\lambda}\left(\frac{a}{m}\right)^m N^2 \leq \frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.93}$$

$$\sigma_1(\boldsymbol{E}_{21})^2\sigma_1((\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-1}) \leq 16^2\frac{a^2N^2}{\lambda}(\frac{a}{m})^{2m} \leq \frac{1}{2}\frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.94}$$

$$\sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}})^2\sigma_1(\boldsymbol{E}_{22}) \leq 16\frac{\rho^2 n^3 a}{\lambda^2}\left(\frac{a}{m}\right)^m \leq 16\frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.95}$$

$$2\sigma_1(\boldsymbol{E}_{21})\sigma_1(\boldsymbol{E}_{22})\sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}}) \leq 16a\left(\frac{a}{m}\right)^m N\frac{4N}{\lambda^2}\left(\frac{a}{m}\right)^m\frac{N\rho}{l} \tag{A.96}$$

$$\leq 2^9\frac{a^2N^2\rho^2}{\lambda^2}(\frac{a}{m})^{2m} \leq \frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.97}$$

$$\sigma_1(\boldsymbol{E}_{21})^2\sigma_1(\boldsymbol{E}_{22}) \leq 2^{10}\frac{a^2N^3}{\lambda^2}(\frac{a}{m})^{3m} \leq \frac{\rho^2 N^3 a^2}{\lambda^2}\left(\frac{a}{m}\right)^m. \tag{A.98}$$

Lemma 13 *Let $m \geq M := 7 + \max\{\frac{e}{2l^2}, \log(\frac{\rho^2 N^3}{\lambda^2\gamma l^4})\}$. Then*

$$\sigma_1(\boldsymbol{E}_3(m+3)) \leq 10\frac{N^2\rho a}{\lambda^2}\left(\frac{a}{m}\right)^m \tag{A.99}$$

Proof 9 *We start by additively decomposing*

$$\boldsymbol{E}_3 = {}'\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}(\tilde{\boldsymbol{C}}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-1} - {}'\boldsymbol{C}_{\boldsymbol{\phi}}(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-1} \tag{A.100}$$

$$= {}'\boldsymbol{C}_{\boldsymbol{\phi}}\boldsymbol{E}_{22} + \boldsymbol{E}_{21}^T(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-1} + \boldsymbol{E}_{21}^T\boldsymbol{E}_{22} \tag{A.101}$$

and then bounding each summand individually, as $m > M$ implies that

$$\sigma_1({}'\boldsymbol{C}_{\boldsymbol{\phi}}\boldsymbol{E}_{22}) \leq \frac{4N^2\rho}{\lambda^2 l}\left(\frac{a}{m}\right)^m \leq 8\frac{N^2\rho a}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.102}$$

$$\sigma_1(\boldsymbol{E}_{21}^T(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-1}) \leq 16\frac{N}{\lambda}a\left(\frac{a}{m}\right)^m \leq \frac{N^2\rho a}{\lambda^2}\left(\frac{a}{m}\right)^m, \tag{A.103}$$

$$\sigma(\boldsymbol{E}_{21}^T\boldsymbol{E}_{22}) \leq 16Na\left(\frac{a}{m}\right)^m\frac{4N}{\lambda^2}\left(\frac{a}{m}\right)^m = 64\frac{N^2a}{\lambda^2}(\frac{a}{m})^{2m} \leq \frac{N^2\rho a}{\lambda^2}\left(\frac{a}{m}\right)^m. \tag{A.104}$$

Thus, we have shown that $\sigma_1(\boldsymbol{E}_3) \leq 10\frac{N^2\rho a}{\lambda^2}\left(\frac{a}{m}\right)^m$.

To deal with the third term, we define

$$T_1 := \left|\left|(\boldsymbol{f}-\tilde{\boldsymbol{D}}\boldsymbol{x})^T(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\tilde{\boldsymbol{D}}\boldsymbol{x}) - (\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})^T(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right|, \tag{A.105}$$

$$T_2 := \left|\left|(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})^T(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x}) - (\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})^T(\boldsymbol{A}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right|. \tag{A.106}$$

Using the triangle inequality, we can observe that

$$\left|\left|(\boldsymbol{f}-\tilde{\boldsymbol{D}}\boldsymbol{x})^T(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\tilde{\boldsymbol{D}}\boldsymbol{x}) - (\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})^T(\boldsymbol{A}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right| \leq T_1+T_2. \tag{A.107}$$

The following two lemmas bound T_1 and T_2, as required for the proof of Theorem 4.

Lemma 14 *Let $m \geq M := 7+\max\left\{\frac{e}{2l^2}, \log\left(\frac{\rho^2n^3}{\lambda^2\gamma l^4}\right)\right\}$. Then, for a quadrature scheme of order $m+3$, it holds that*

$$T_2 \leq \epsilon_{32}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})^T(\boldsymbol{A}+\gamma\boldsymbol{I})^{-1}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x}), \tag{A.108}$$

where

$$\epsilon_{32} = 40\frac{\rho^2N^3a^2}{\lambda^2\gamma}\left(\frac{a}{m}\right)^m. \tag{A.109}$$

Proof 10 *The proof of this lemma is identical to the proof of Lemma 11, using $\boldsymbol{A}+\gamma\boldsymbol{I}$ instead of $\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I}$, $\boldsymbol{E}_2$ instead of $\boldsymbol{E}_1$ and $(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})$ instead of $\boldsymbol{x}$.*

Lemma 15 *Let* $m \geq M := 7 + \max\left\{\frac{e}{2l^2}, \log\left(\frac{\rho^2 n^3}{\lambda^2 \gamma l^4}\right)\right\}$. *Then, fora quadrature scheme of order* $m + 3$, *it holds that*

$$T_1 \leq \epsilon_{31}(\boldsymbol{f} - \boldsymbol{D}\boldsymbol{x})^T(\boldsymbol{A} + \gamma \boldsymbol{I})^{-1}(\boldsymbol{f} - \boldsymbol{D}\boldsymbol{x}) + \epsilon_{31}\boldsymbol{x}^T(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{x} \quad \text{(A.110)}$$

where $\epsilon_{31} = 30\frac{N^{\frac{5}{2}}\rho^{\frac{3}{2}}a}{\lambda^2\gamma^{\frac{1}{2}}}\left(\frac{a}{m}\right)^m$.

Proof 11 *We know that*

$$\sigma_1((\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-\frac{1}{2}})\sigma_1(\boldsymbol{E}_3)\sigma_1((\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{\frac{1}{2}}) \leq 10\frac{N^2\rho a}{\lambda^2}\left(\frac{a}{m}\right)^m\left(\frac{\rho N + \lambda}{\gamma}\right)^{\frac{1}{2}} \quad \text{(A.111)}$$

$$\leq 15\frac{N^{\frac{5}{2}}\rho^{\frac{3}{2}}a}{\lambda^2\gamma^{\frac{1}{2}}}\left(\frac{a}{m}\right)^m \quad \text{(A.112)}$$

$$= \epsilon_{31}/2. \quad \text{(A.113)}$$

Furthermore,

$$T_1 = \left|\left|2(\boldsymbol{E}_3\boldsymbol{x})^T(\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-1}(\boldsymbol{f} - \boldsymbol{D}\boldsymbol{x}) + (\boldsymbol{E}_3\boldsymbol{x})^T(\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-1}(\boldsymbol{E}_3\boldsymbol{x})\right|\right| \quad \text{(A.114)}$$

$$\leq 2\,||\boldsymbol{E}_3\boldsymbol{x}||\,\sigma_1\left((\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-\frac{1}{2}}\right)\left|\left|(\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{f} - \boldsymbol{D}\boldsymbol{x})\right|\right| + \sigma_1\left((\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-1}\right)||\boldsymbol{E}_3\boldsymbol{x}||^2 \quad \text{(A.115)}$$

For the second summand of Equation (A.115), *observe that*

$$||\boldsymbol{E}_3\boldsymbol{x}|| = \left|\left|\boldsymbol{E}_3(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{\frac{1}{2}}(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right| \quad \text{(A.116)}$$

$$\leq \sigma_1(\boldsymbol{E}_3)\sigma_1\left((\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{\frac{1}{2}}\right)\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right| \quad \text{(A.117)}$$

implies

$$\sigma_1\left((\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-1}\right)||\boldsymbol{E}_3\boldsymbol{x}||^2 \quad \text{(A.118)}$$

$$\leq \sigma_1\left((\tilde{\boldsymbol{A}} + \gamma \boldsymbol{I})^{-1}\right)\sigma_1\left(\boldsymbol{E}_3\right)^2\sigma_1\left((\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{\frac{1}{2}}\right)^2\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|^2 \quad \text{(A.119)}$$

$$\leq \frac{\epsilon_{31}^2}{4}\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|^2 \quad \text{(A.120)}$$

$$\leq \frac{\epsilon_{31}}{2}\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}} + \lambda \boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|^2. \quad \text{(A.121)}$$

For the first summand of Equation (A.115)*, observe that*

$$
\begin{aligned}
&2\,||\boldsymbol{E}_3\boldsymbol{x}||\,\sigma_1\left((\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}\right)\left|\left|(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}(f(\boldsymbol{x},\boldsymbol{\theta})-\boldsymbol{D}\boldsymbol{x})\right|\right| \\
&= 2\sigma_1\left((\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}\right)\left|\left|\boldsymbol{E}_3(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{\frac{1}{2}}(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|\left|\left|(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right| \\
&\leq 2\sigma_1\left((\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}\right)\sigma_1\left(\boldsymbol{E}_3\right)\sigma_1\left((\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{\frac{1}{2}}\right)\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|\left|\left|(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{f}-\boldsymbol{I}\right.\right. \\
&\leq \frac{\epsilon_{31}}{2}\left(\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|^2+\left|\left|(\tilde{\boldsymbol{A}}+\gamma\boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right|^2\right) \\
&\leq \frac{\epsilon_{31}}{2}\left(\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|^2+(1+\epsilon_{32})\left|\left|(\boldsymbol{A}+\gamma\boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right|^2\right) \\
&\leq \frac{\epsilon_{31}}{2}\left|\left|(\boldsymbol{C}_{\boldsymbol{\phi}}+\lambda\boldsymbol{I})^{-\frac{1}{2}}\boldsymbol{x}\right|\right|^2+\epsilon_{31}\left|\left|(\boldsymbol{A}+\gamma\boldsymbol{I})^{-\frac{1}{2}}(\boldsymbol{f}-\boldsymbol{D}\boldsymbol{x})\right|\right|^2
\end{aligned}
$$

where the second to last inequality comes from Equation (A.108)*, while the last follows from the fact that* $\epsilon_{32} \leq 1$ *for* $m \geq M$*, concluding the proof of this lemma.*

A.2.7.3 *Proof of Theorem 4*

Combining the results of Lemmas 11, 15and 14, proving Theorem 4 is straight forward.

Consider $m \geq M := 7 + \max\left\{\frac{e}{2l^2}, \log\left(\frac{\rho^2 N^3}{\lambda^2\gamma l^4}\right)\right\}$ and let $m' = m + 3$. Observing the facts that that $\epsilon_1 \leq \epsilon_{32}$ and $\epsilon_{31} \leq \epsilon_{32}$, we get

$$
\begin{aligned}
\frac{\left|\mathcal{R}_{\lambda\gamma\boldsymbol{\phi}}(x,\boldsymbol{\theta})-\tilde{\mathcal{R}}_{\lambda\gamma\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\theta})\right|}{\mathcal{R}_{\lambda\gamma\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{\theta})} &\leq 2\epsilon_{32} && \text{(A.122)}\\
&= 80\frac{\rho^2 N^3 a^2}{\lambda^2\gamma}\left(\frac{a}{m}\right)^m && \text{(A.123)}\\
&\leq 50\frac{\rho^2 N^3}{\lambda^2\gamma l^4}\left(\frac{a}{m}\right)^m. && \text{(A.124)}
\end{aligned}
$$

In order to make that smaller than ϵ it suffices $m \geq \max\left\{\frac{e}{2l^2}, \log\left(50\frac{\rho^2 N^3}{\lambda^2\gamma l^4\epsilon}\right)\right\}$. Thus, we retrieve the desired error bounds for a quadrature scheme of order m', where

$$
m' = 10 + \max\left\{\frac{e}{2l^2}, \log\left(\frac{\rho^2 N^3}{\lambda^2\gamma l^4\epsilon}\right)\right\}. \qquad \text{(A.125)}
$$

A.2.8 *Experimental Setups*

Here, we will give a brief overview of the experimental setups we used throughout this paper.

A.2.8.1 *Basic Definitions*

The trajectory RMSE has proven to be an efficient metric to evaluate the quality of a parameter inference scheme, especially in the context of non-identifiable systems. Here, we restate its definition as used by Wenk et al. [Wen+20].

Definition 2 (Trajectory RMSE) *Let $\hat{\boldsymbol{\theta}}$ be the parameters estimated by an algorithm. Let $\mathbf{t}$ be the vector collecting the observation times. Define $\tilde{\mathbf{x}}(t)$ as the trajectory one obtains by integrating the ODEs using the estimated parameters, but the true initial value, i.e.*

$$\tilde{x}(0) = \mathbf{x}^*(0) \tag{A.126}$$

$$\tilde{x}(t) = \int_0^t f(\tilde{x}(s), \hat{\boldsymbol{\theta}}) ds \tag{A.127}$$

and define $\tilde{\mathbf{x}}$ element-wise as its evaluation at observation times $\mathbf{t}$, i.e. $\tilde{\mathbf{x}}_i = \tilde{x}(t_i)$. The trajectory RMSE is then defined as

$$tRMSE := \frac{1}{\sqrt{N}} ||\tilde{\mathbf{x}} - \mathbf{x}||_2 \tag{A.128}$$

where $||.||_2$ denotes the standard Euclidean 2-norm.

Additionally, we would like to restate the definition of the signal-to-noise ratio, as it was used in our work to create the observation noise for the Quadrocopter system.

Definition 3 (Signal to Noise Ratio) *Let $x(t)$ be a time-continuous signal for a closed time interval. Let σ_x^2 denote its variance across time. Furthermore, let σ^2 be the variance of an additive, Gaussian noise signal. Then we define the SNR as the ration of these two variances, i.e.*

$$SNR = \frac{\sigma_x^2}{\sigma^2}. \tag{A.129}$$

A.2.8.2 *Lotka Volterra*

$$\begin{aligned}\dot{x}_1(t) = &\quad \theta_1 x_1(t) - \theta_2 x_1(t) x_2(t) \\ \dot{x}_2(t) = &\quad -\theta_3 x_2(t) + \theta_4 x_1(t) x_2(t).\end{aligned} \tag{A.130}$$

The Lotka Volterra system [Lot32] has become a widely used benchmarking system. Due to its locally linear dynamics [GBB17] and relatively tame trajectories, it is a system many algorithms can solve. We follow the standard setting in the literature and use $\boldsymbol{\theta} = [2, 1, 4, 1]$ and $\mathbf{x}(0) = [5, 3]$ to generate trajectories over the time interval $[0, 2]$. The dynamics are shown in Equation (A.130).

A.2.8.3 *Protein Transduction*

$$\begin{aligned}\dot{S} &= -\theta_1 S - \theta_2 S R + \theta_3 R_S \\ \dot{dS} &= \theta_1 S \\ \dot{R} &= -\theta_2 S R + \theta_3 R_S + \theta_5 \frac{R_{pp}}{\theta_6 + R_{pp}} \\ \dot{R}_S &= \theta_2 S R - \theta_3 R_S - \theta_4 R_S \\ \dot{R}_{pp} &= \theta_4 R_S - \theta_5 \frac{R_{pp}}{\theta_6 + R_{pp}}\end{aligned} \tag{A.131}$$

A more challenging system was introduced by Vyshemirsky and Girolami [VG07]. Its nonlinear terms and non-stationarity introduce interesting challenges for many collocation methods. We follow the standard setting in the literature and use $\boldsymbol{\theta} = [0.07, 0.6, 0.05, 0.3, 0.017, 0.3]$ and $\mathbf{x}(0) = [1, 0, 1, 0, 0]$, but change the time interval to generate trajectories over the time interval $[0, 50]$, since they stay pretty much constant for $t > 50$. The dynamics are shown in Equation (A.131).

A.2.8.4 *Lorenz 63*

$$\dot{x} = \theta_0 (y - x) \tag{A.132}$$

$$\dot{y} = x(\theta_1 - z) - y \tag{A.133}$$

$$\dot{z} = xy - \theta_2 z \tag{A.134}$$

The Lorenz 63 system was introduced by Lorenz [Lor63] to model atmospheric flows. It is an optimal test bed for parameter inference algorithms,

as it exhibits chaotic behavior for the parameter settings we chose. Working with chaotic dynamics is notoriously challenging due to high sensitivity to parameter changes and the presence of many local optima. We follow standard literature and use $\boldsymbol{\theta} = [10, 28, 8/3]$ and $\mathbf{x}(0) = [1, 1, 1]$ to generate trajectories over the time interval $[0, 1]$. The dynamics are shown in Equation (A.134).

A.2.8.5 *Quadrocopter*

$$
\begin{aligned}
\dot{x}_0 &= -g\sin(x_7) + x_5x_1 - x_4x_2 \\
\dot{x}_1 &= g\sin(x_6)\cos(x_7) - x_0x_5 + x_2x_3 \\
\dot{x}_2 &= -\frac{u_0 + u_1 + u_2 + u_3}{\theta_0} + g\cos(x_6)\cos(x_7) + x_0x_4 - \theta_4x_1 \\
\dot{x}_3 &= \frac{1}{\theta_1}(\theta_5(-u_0 + u_1 + u_2 - u_3)) + (\theta_2 - \theta_3(\theta_2 + \theta_1)))x_4x_5) \\
\dot{x}_4 &= \frac{1}{\theta_2}(\theta_4(u_0 - u_1 + u_3 - u_4) + (\theta_3(\theta_2 + \theta_1) - \theta_1)x_3x_5) \\
\dot{x}_5 &= \frac{(\theta_1 - \theta_2)x_3x_4}{\theta_3(\theta_2 + \theta_1)} \\
\dot{x}_6 &= x_3 + (x_4\sin(x_6) + \frac{x_5\cos(x_6)\sin(x_7)}{\cos(x_7)} \\
\dot{x}_7 &= x_4\cos(x_6) - x_5\sin(x_6) \\
\dot{x}_8 &= \frac{x_4\sin(x_6) + x_5\cos(x_6)}{\cos(x_7)} \\
\dot{x}_9 &= \cos(x_7)\cos(x_8)x_0 + (-\cos(x_6)\sin(x_8) + \sin(x_6)\sin(x_7)\cos(x_8))x_1 \\
&\quad + (\sin(x_6)\sin(x_8) + \cos(x_6)\sin(x_7)\cos(x_8))x_2 \\
\dot{x}_{10} &= \cos(x_7)\sin(x_8)x_0 + (\cos(x_6)\cos(x_8) + \sin(x_6)\sin(x_7)\sin(x_8))x_1 \\
&\quad + (\cos(x_6)\sin(x_7)\sin(x_8) - \sin(x_6)\cos(x_8))x_2 \\
\dot{x}_{11} &= \sin(x_7)x_0 - \sin(x_6)\cos(x_7)x_1 - \cos(x_6)\cos(x_7)x_2
\end{aligned}
\tag{A.135}
$$

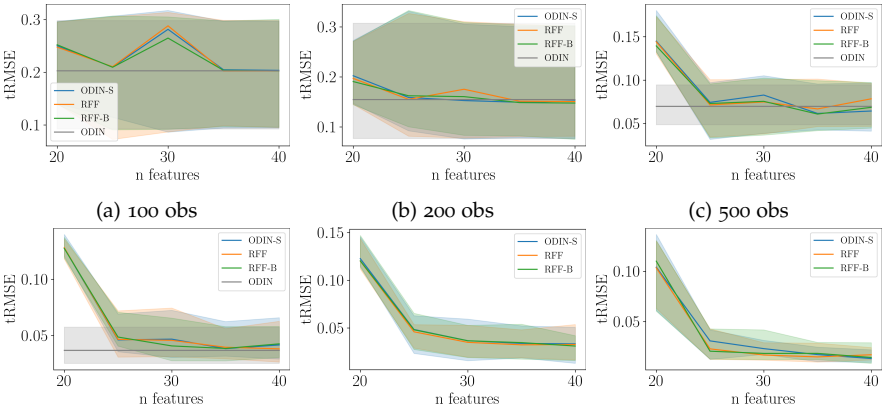
As an ultimate benchmark, we introduce a parametric model describing the dynamics of a 6DOF quadrocopter, shown in Equation (A.135). Its strongly nonlinear dynamics and the presence of inputs make it a formidable challenge. The states of this system are representing the linear velocities (x_0, x_1, x_2), the angular velocities (x_3, x_4, x_5), the angles (x_6, x_7, x_8) and the position (x_9, x_{10}, x_{11}) of the quadrocopter. The four inputs represent the forces applied at the four different propellers. While in prin-

ciple any input commands could be incorporated, we keep the inputs constant at $u = [0.248, 0.2475, 0.24775, 0.24775]$. This input leads to interesting nonstationary climbing, pitching and rolling behavior. We use $\boldsymbol{\theta} = [0.1, 0.00062, 0.00113, 0.9, 0.114, 0.0825, 9.85]$ and $x_i(0) = 0$ for $i = 0 \ldots 11$ to generate trajectories over the time interval $[0, 15]$.

A.2.9 *Additional Empirical Evaluation SLEIPNIR*

A.2.10 *tRMSE vs Features*

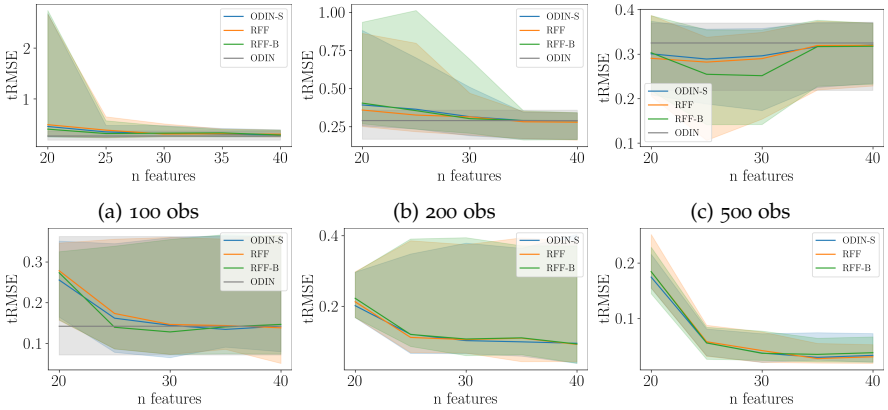
A.2.10.1 *Lotka Volterra*



(a) 100 obs (b) 200 obs (c) 500 obs

(d) 1000 obs (e) 2000 obs (f) 5000 obs

FIGURE A.21: tRMSE vs features for the Lotka Volterra system using additive observation noise with $\sigma^2 = 0.1$.

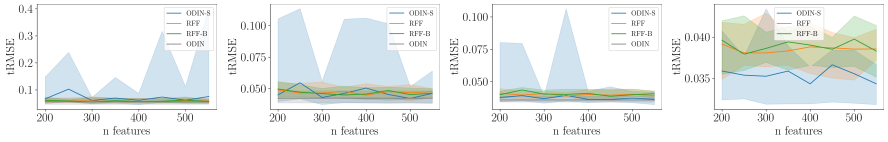


(a) 100 obs (b) 200 obs (c) 500 obs

(d) 1000 obs (e) 2000 obs (f) 5000 obs

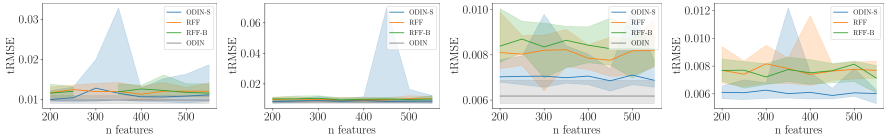
FIGURE A.22: tRMSE vs features for the Lotka Volterra system using additive observation noise with $\sigma^2 = 0.5$.

A.2.10.2 *Protein Transduction*



(a) 500 obs (b) 1000 obs (c) 2000 obs (d) 3000 obs

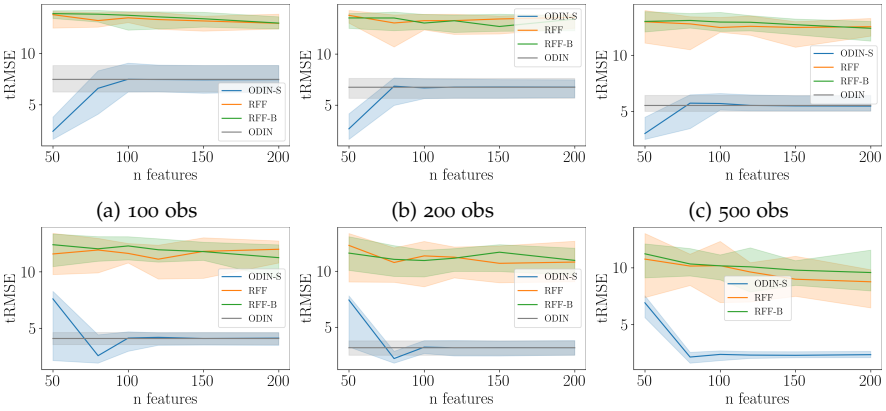
FIGURE A.23: tRMSE vs features for the Protein Transduction system using additive observation noise with $\sigma^2 = 0.01$.



(a) 500 obs (b) 1000 obs (c) 2000 obs (d) 3000 obs

FIGURE A.24: tRMSE vs features for the Protein Transduction system using additive observation noise with $\sigma^2 = 0.0001$.

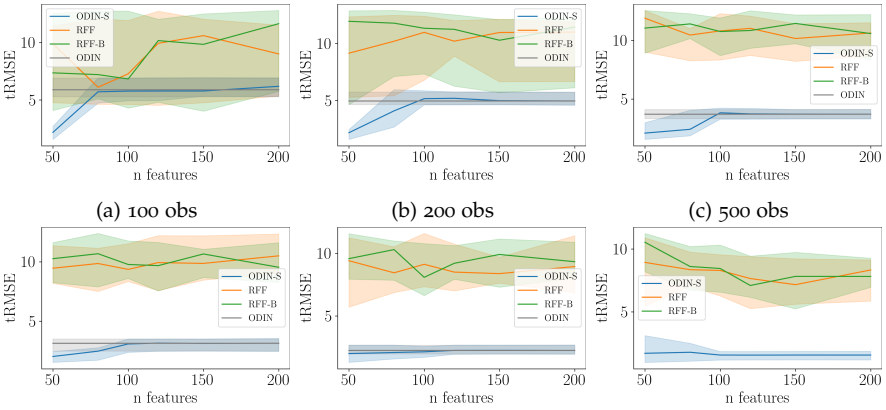
A.2.10.3 *Lorenz*



(a) 100 obs (b) 200 obs (c) 500 obs

(d) 1000 obs (e) 2000 obs (f) 5000 obs

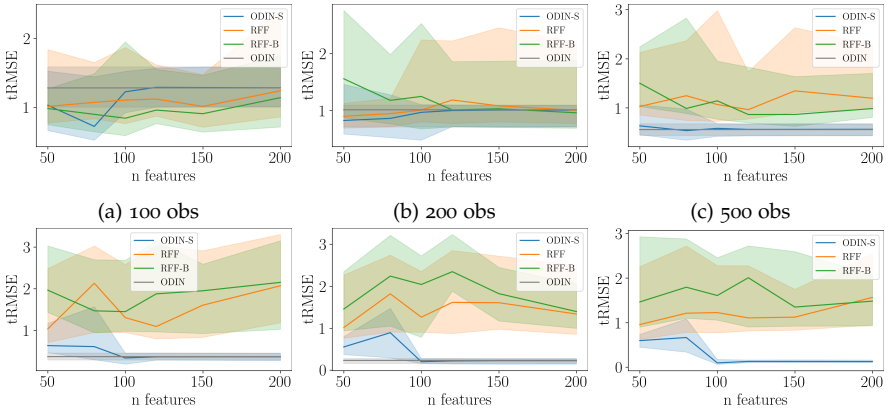
FIGURE A.25: tRMSE vs features for the Lorenz system with noise created using a signal-to-noise ratio of 5.



(a) 100 obs (b) 200 obs (c) 500 obs

(d) 1000 obs (e) 2000 obs (f) 5000 obs

FIGURE A.26: tRMSE vs features for the Lorenz system with noise created using a signal-to-noise ratio of 10.



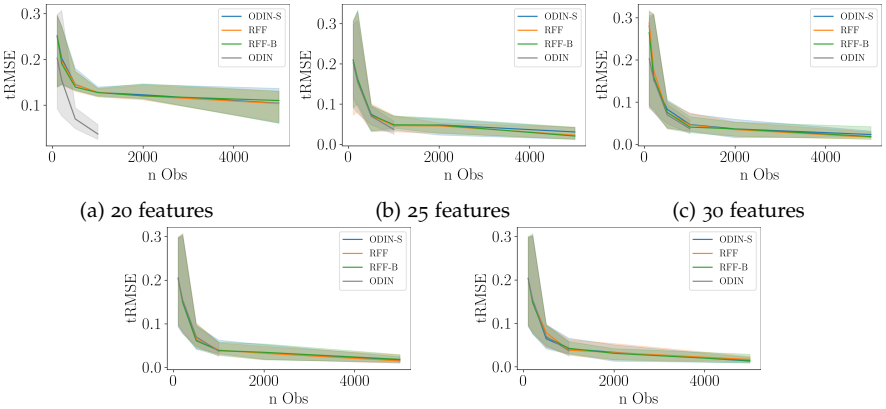
(a) 100 obs (b) 200 obs (c) 500 obs

(d) 1000 obs (e) 2000 obs (f) 5000 obs

FIGURE A.27: tRMSE vs features for the Lorenz system with noise created using a signal-to-noise ratio of 100.

A.2.11 *Learning Curves*

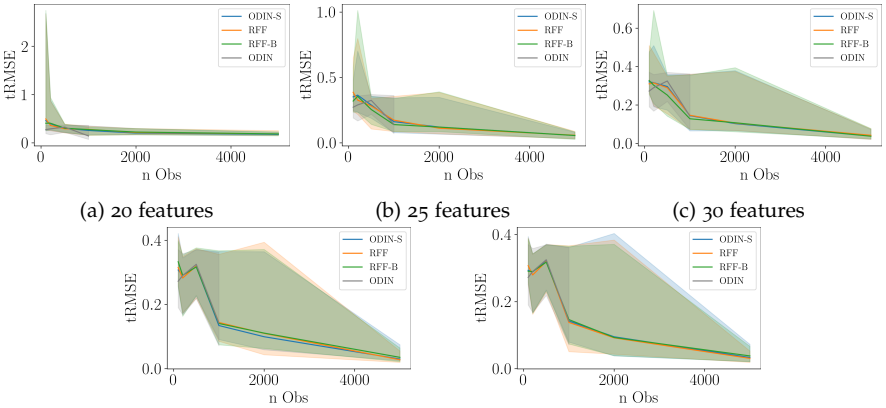
A.2.11.1 *Lotka Volterra*



(a) 20 features (b) 25 features (c) 30 features

(d) 35 features (e) 40 features

FIGURE A.28: tRMSE vs number of observations for the Lotka Volterra system with additive noise with $\sigma^2 = 0.1$.

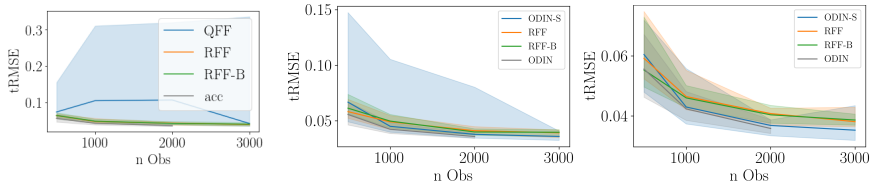


(a) 20 features (b) 25 features (c) 30 features

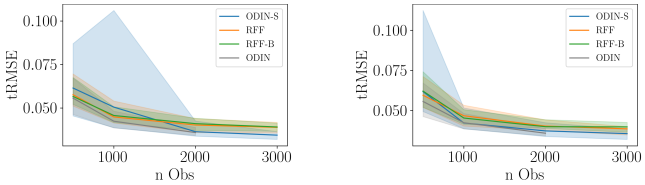
(d) 30 features (e) 35 features

FIGURE A.29: tRMSE vs number of observations for the Lotka Volterra system with additive noise with $\sigma^2 = 0.5$.

A.2.11.2 *Protein Transduction*

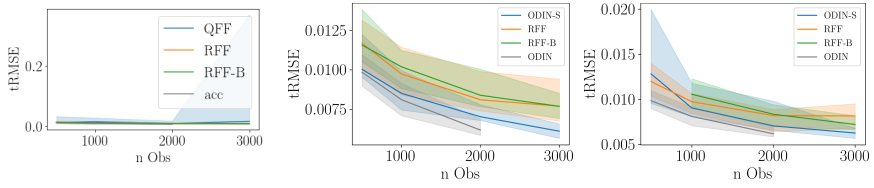


(a) 100 features (b) 200 features (c) 300 features

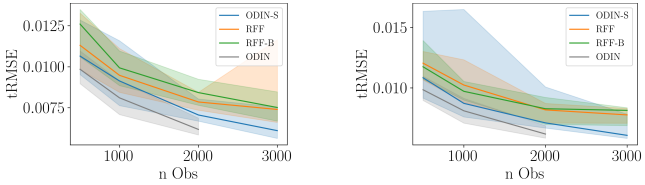


(d) 400 features (e) 500 features

FIGURE A.30: tRMSE vs number of observations for the Protein Transduction system using additive Gaussian noise with $\sigma^2 = 0.01$.



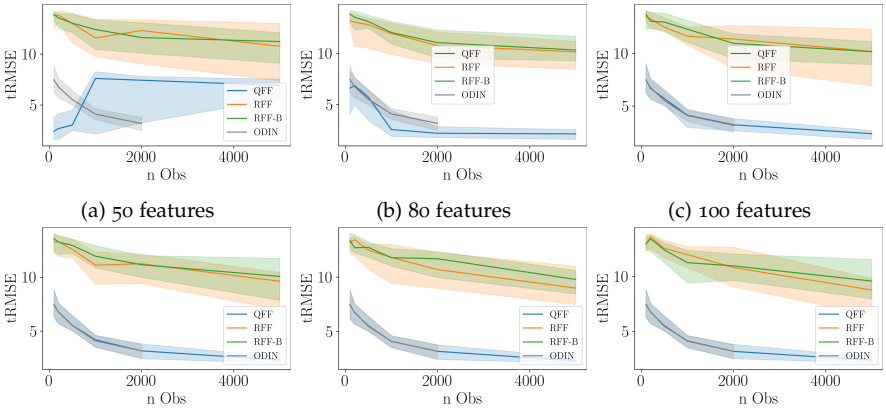
(a) 100 features (b) 200 features (c) 300 features



(d) 400 features (e) 500 features

FIGURE A.31: tRMSE vs number of observations for the Protein Transduction system using additive Gaussian noise with $\sigma^2 = 0.0001$.

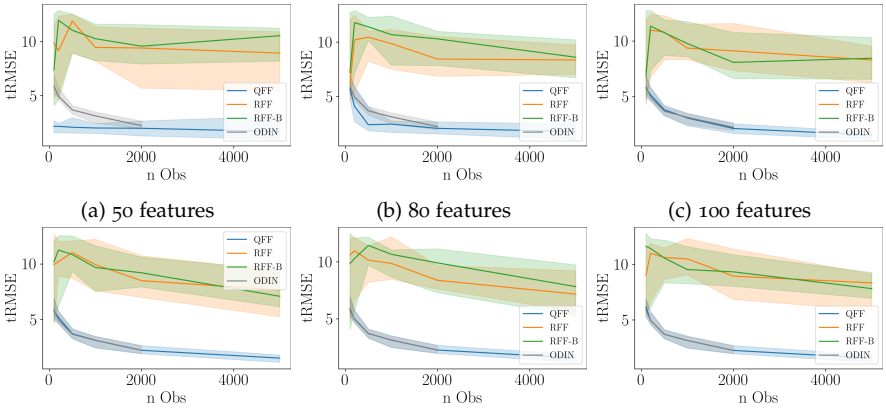
A.2.11.3 *Lorenz*



(a) 50 features (b) 80 features (c) 100 features

(d) 120 features (e) 150 features (f) 200 features

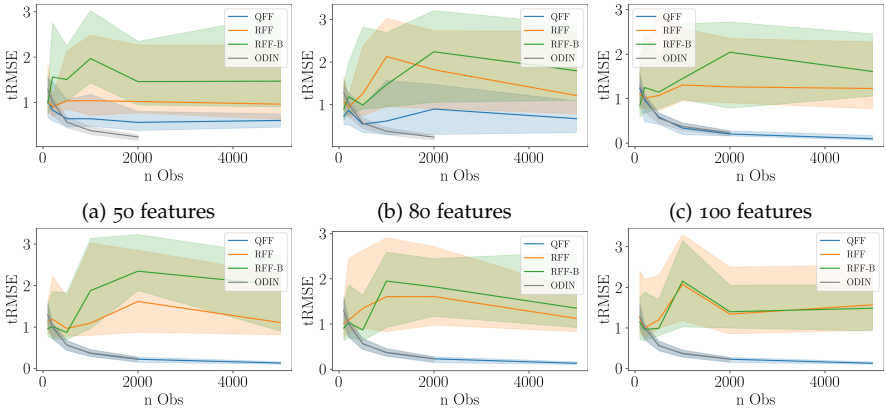
FIGURE A.32: tRMSE vs number of observations for the Lorenz system with noise created using a signal-to-noise ratio of 5.



(a) 50 features (b) 80 features (c) 100 features

(d) 120 features (e) 150 features (f) 200 features

FIGURE A.33: tRMSE vs number of observations for the Lorenz system with noise created using a signal-to-noise ratio of 10.

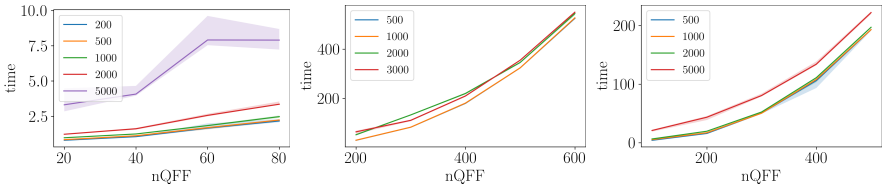


(a) 50 features (b) 80 features (c) 100 features

(d) 120 features (e) 150 features (f) 200 features

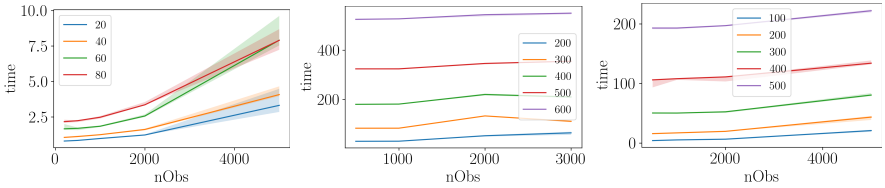
FIGURE A.34: tRMSE vs number of observations for the Lorenz system with noise created using a signal-to-noise ratio of 100.

A.2.11.4 *Run Time*



(a) LV (b) PT (c) Lorenz

FIGURE A.35: Run time per iteration in ms vs number of features for different numbers of observations. As expected from theoretical analysis, the run time per iteration scales approximately cubic.



(a) LV (b) PT (c) Lorenz

FIGURE A.36: Run time per iteration in ms vs number of observations for different numbers of Fourier features. As expected from theoretical analysis, the run time per iteration scales approximately linear, even though there is a strong bias term.

A.3 APPENDIX TO DGM

A.3.1 *Dataset description*

In this section, we describe the datasets we use in our experiments.

A.3.1.1 *Lotka Volterra*

The two dimensional Lotka Volterra system is governed by the parametric differential equations

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y,$$

where we selected $(\alpha, \beta, \gamma, \delta) = (1, 1, 1, 1)$. These equations were numerically integrated to obtain a ground truth, where the initial conditions and observation times depend on the dataset. All observations were then created by adding additive, i.i.d. noise, distributed according to a normal distribution $\mathcal{N}\left(0, 0.1^2\right)$. This is the same system that was already introduced in Section 2.7, but with different ground truth settings to obtain a slightly more interesting trajectory shape.

LV 1 consists of one trajectory starting from initial condition $(1, 2)$. The trajectory is observed at 100 equidistant time points from the interval $(0, 10)$.

LV 100 consists of 100 trajectories. Initial conditions for these trajectories are located on a grid, i.e.,

$$\left\{ \left(\frac{1}{2} + \frac{i}{9}, \frac{1}{2} + \frac{j}{9} \right) \middle| i \in \{0, \ldots, 9\}, j \in \{0, \ldots, 9\} \right\}.$$

Each trajectory is then observed at 5 equidistant time points from the interval $(0, 10)$, which leads to a total of 500 observations.

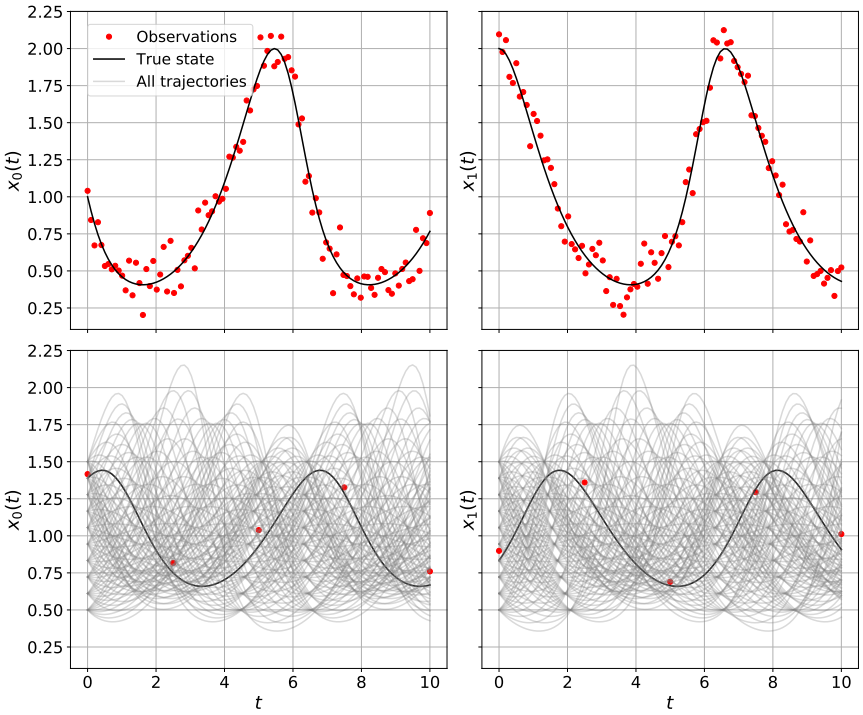


FIGURE A.37: The first row represents the true states and all observations of LV 1 with random seed 0. In the second row we plot all ground truth trajectories from the dataset LV 100. One particular trajectory is highlighted in black, together with the corresponding observations of that trajectory (red dots).

To test generalization, we created 10 new trajectories. The initial conditions of these trajectories were obtained by sampling uniformly at random on $[0.5, 1.5]^2$. To evaluate the log likelihood, we used 100 equidistant time points from the interval $(0, 10)$.

A.3.1.2 *Lorenz*

The 3 dimensional, chaotic Lorenz system is governed by the parametric differential equations

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x)\\ \frac{dy}{dt} &= x(\rho - z) - y\\ \frac{dz}{dt} &= xy - \tau y,\end{aligned}$$

where we selected $(\sigma, \rho, \tau) = (10, 28, 8/3)$. These equations were numerically integrated to obtain a ground truth, where the initial conditions and observation times depend on the dataset. All observations were then created by adding additive, i.i.d. noise, distributed according to a normal distribution $\mathcal{N}(0, 1)$. This is a system similar to the Lorenz system we used in the scaling experiments in Section 3.3.3, but fixed to three dimensions and with a different parameter structure. However, they are closely related.

LO 1 consists of one trajectory starting from initial condition $(-2.5, 2.5, 2.5)$. The trajectory is observed at 100 equidistant time points from the interval $(0, 1)$.

LO 125 consists of 125 trajectories. Initial conditions for these trajectories are located on a grid, i.e.,

$$\left\{ \left(-5 + \frac{5i}{2}, -5 + \frac{5j}{2}, -5 + \frac{5k}{2} \right) \middle| i \in \{0, \dots, 4\}, j \in \{0, \dots, 4\}, k \in \{0, \dots, 4\} \right\}.$$

Each trajectory is then observed on 10 equidistant time points from the interval $(0, 1)$, which leads to a total of 1250 observations.

To test generalization, we created 10 new trajectories. The initial conditions of these trajectories were obtained by sampling uniformly at random on $[-5, 5]^3$. To evaluate the log likelihood, we used 100 equidistant time points from the interval $(0, 1)$.

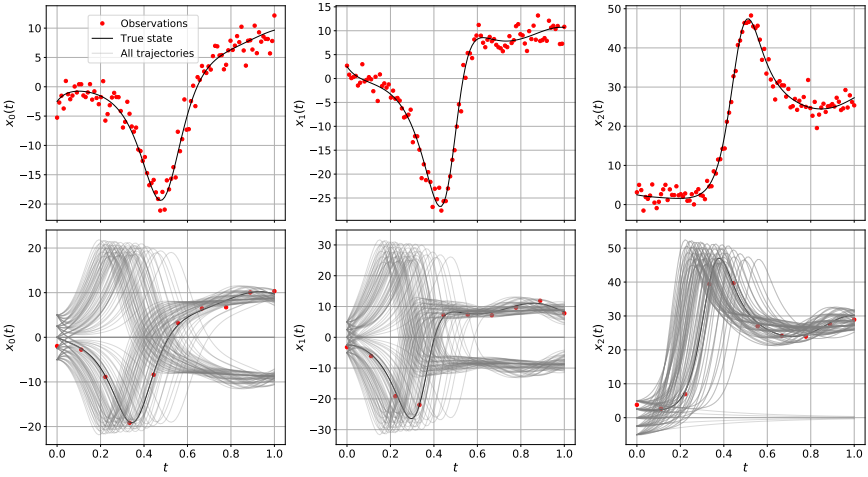


FIGURE A.38: The first row represents the true states and all observations of LO 1 with random seed 0. In the second row we plot all ground truth trajectories from the dataset LO 125. One particular trajectory is highlighted in black, together with the corresponding observations of that trajectory (red dots).

A.3.1.3 *Double Pendulum*

The 4 dimensional Double pendulum system is governed by the parametric differential equations where we selected $(g, m, l) = (9.81, 1, 1)$. In these equations, θ_1 and θ_2 represent the offset angles, while p_{θ_1} and p_{θ_1} represent the momentum of the upper and lower pendulum. These equations were numerically integrated to obtain a ground truth, where the initial conditions and observation times depend on the dataset.

$$
\begin{aligned}
\dot{\theta}_1 &= \frac{6}{ml^2}\frac{2p_{\theta_1} - 3\cos(\theta_1 - \theta_2)p_{\theta_2}}{16 - 9\cos^2(\theta_1 - \theta_2)} \\
\dot{\theta}_2 &= \frac{6}{ml^2}\frac{8p_{\theta_2} - 3\cos(\theta_1 - \theta_2)p_{\theta_1}}{16 - 9\cos^2(\theta_1 - \theta_2)}. \\
\dot{p}_{\theta_1} &= -\tfrac{1}{2}ml^2\left(\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) + 3\frac{g}{l}\sin\theta_1\right) \\
\dot{p}_{\theta_2} &= -\tfrac{1}{2}ml^2\left(-\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) + \frac{g}{l}\sin\theta_2\right),
\end{aligned}
$$

All observations were then created by adding additive, i.i.d. noise, distributed according to a normal distribution $\mathcal{N}\left(0, 0.1^2\right)$.

DP 1 consist of one trajectory starting from initial condition $(-\pi/6, -\pi/6, 0, 0)$. The trajectory is observed at 100 equidistant time points from the interval $(0, 1)$.

DP 100 consists of 100 trajectories. Initial conditions for these trajectories are located on a grid, i.e.,

$$\left\{ \left(-\frac{\pi}{6} + \frac{\pi i}{27}, -\frac{\pi}{6} + \frac{\pi j}{27}, 0, 0 \right) \middle| i \in \{0, \ldots, 9\}, j \in \{0, \ldots, 9\} \right\}.$$

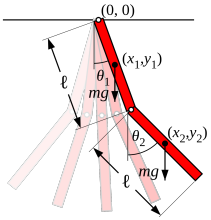


FIGURE A.39: Double Pendulum where both rods have equal length and mass.

Each trajectory is then observed at 5 equidistant time points from the interval $(0, 1)$, which leads to a total of 500 observations.

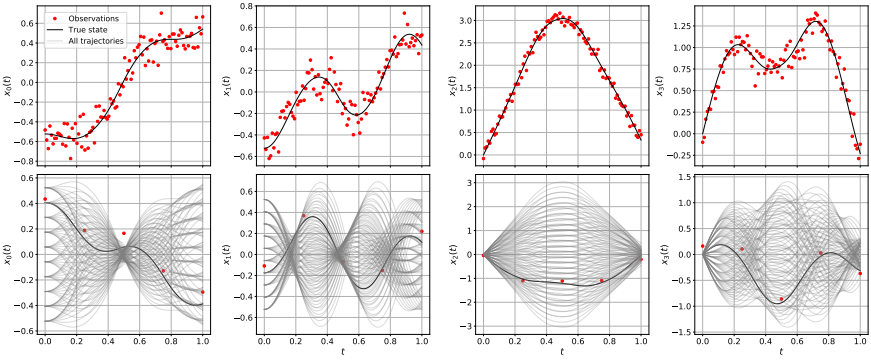


FIGURE A.40: The first row represents the true states and all observations of DP 1 with random seed 0. In the second row we plot all ground truth trajectories from the dataset DP 100. One particular trajectory is highlighted in black, together with the corresponding observations of that trajectory (red dots).

To test generalization, we created 10 new trajectories. The initial conditions of these trajectories were obtained by sampling uniformly at random on $[-\frac{\pi}{6}, \frac{\pi}{6}]^2 \times \{0\}^2$. To evaluate the log likelihood, we used 100 equidistant time points from the interval $(0, 1)$.

A.3.1.4 *Quadrocopter*

The 12 dimensional Quadrocopter system is governed by the parametric differential equations

$$
\begin{aligned}
\dot{u} &= -g\sin(\theta) + rv - qw \\
\dot{v} &= g\sin(\phi)\cos(\theta) - ru + pw \\
\dot{w} &= -F_z/m + g\cos(\phi)\cos(\theta) + qu - pv \\
\dot{p} &= \left(L + (I_{yy} - I_{zz})qr\right)/I_{xx} \\
\dot{q} &= \left(M + (I_{zz} - I_{xx})pr\right)/I_{yy} \\
\dot{r} &= (I_{xx} - I_{yy})pq/I_{zz} \\
\dot{\phi} &= p + (q\sin(\phi) + r\cos(\phi))\tan(\theta) \\
\dot{\theta} &= q\cos(\phi) - r\sin(\phi) \\
\dot{\psi} &= (q\sin(\phi) + r\cos(\phi))\sec(\theta) \\
\dot{x} &= \cos(\theta)\cos(\psi)u + (-\cos(\phi)\sin(\psi) + \sin(\phi)\sin(\theta)\cos(\psi))v + \\
&\quad + (\sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\phi))w \\
\dot{y} &= \cos(\theta)\sin(\psi)u + (\cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi))v + \\
&\quad + (-\sin(\phi)\cos(\psi) + \cos(\phi)\sin(\theta)\sin(\phi))w \\
\dot{z} &= \sin(\theta)u - \sin(\phi)\cos(\theta)v - \cos(\phi)\cos(\theta)w,
\end{aligned}
$$

where

$$
\begin{aligned}
F_z &= F_1 + F_2 + F_3 + F_4 \\
L &= (F_2 + F_3)d_y - (F_1 + F_4)d_x \\
M &= (F_1 + F_3)d_x - (F_2 + F_4)d_x.
\end{aligned}
$$

We fixed the input control forces to $(F_1, F_2, F_3, F_4) = (0.496, 0.495, 0.4955, 0.4955)$ (not to be inferred) and selected $(m, I_{xx}, I_{yy}, I_{zz}, d_x, d_y, g) = (0.1, 0.62, 1.13, 0.9, 0.114, 0.08$ These equations were numerically integrated to obtain a ground truth, where the initial conditions and observation times depend on the dataset. All observations were then created by adding additive, i.i.d. noise, distributed according to a normal distribution $\mathcal{N}(0, \Sigma)$, where

$$
\Sigma = \mathrm{diag}(1, 1, 1, 0.1, 0.1, 0.1, 1, 0.1, 0.1, 5, 5, 5).
$$

QU 1 consists of one trajectory starting from initial condition

$$
\boldsymbol{x}(0) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). \tag{A.136}
$$

The trajectory is observed at 100 equidistant time points from the interval $(0, 10)$.

QU 64 consists of 64 trajectories. Initial conditions for these trajectories are located on a grid, i.e.,

$$\left\{ \left(0,0,0,0,0,0,-\frac{\pi}{18}+\frac{\pi i}{27},-\frac{\pi}{18}+\frac{\pi j}{27},-\frac{\pi}{18}+\frac{\pi k}{27},0,0,0\right) \middle| (i,j,k) \in \{0,\dots,4\}^3 \right\}.$$

Each trajectory is then observed at 15 equidistant time points from the interval $(0, 10)$, which leads to a total of 960 observations.

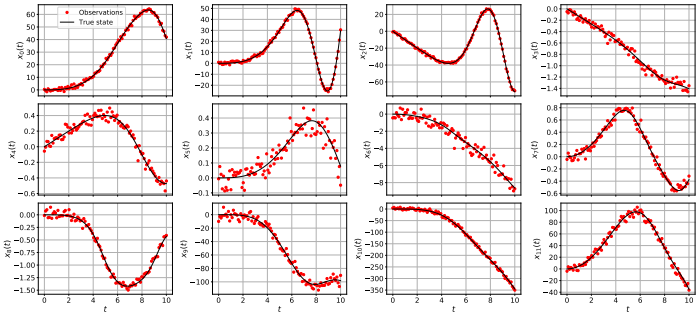


FIGURE A.41: Visualization showing the true states and all observations of QU 1 with random seed 0.

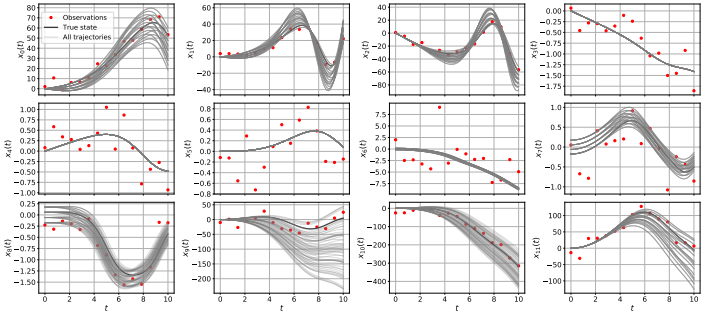


FIGURE A.42: Visualization showing all ground truth trajectories from the dataset QU 64. One particular trajectory is highlighted in black, together with the corresponding observations of that trajectory (red dots).

To test generalization, we created 10 new trajectories. The initial conditions of these trajectories were obtained by sampling uniformly at random

on $\{0\}^6 \times [-\frac{\pi}{18}, \frac{\pi}{18}]^2 \times \{0\}^3$. To evaluate the log likelihood, we used 100 equidistant time points from the interval $(0, 10)$.

A.3.2 *Implementation details of DGM*

In this section we discuss all implementation details of DGM. As described in Section 4.2, DGM consists of a smoother and a dynamics model. At training time, the dynamics model is then used to regularize the smoother via the squared type-2 Wasserstein distance.

A.3.2.1 *Dynamics model*

The role of the dynamics model is to map every state $\boldsymbol{x}$ to a distribution over derivatives, which we decide to parameterize as $\mathcal{N}(\boldsymbol{f}(\boldsymbol{x}, \psi), \boldsymbol{\Sigma}_D(\boldsymbol{x}, \psi))$. In the paper we focused in the case where we do not have any prior knowledge about the dynamics and we model both $\boldsymbol{f}$ and $\boldsymbol{\Sigma}_D$ with neural network. Nevertheless, if some prior knowledge about the dynamics $\boldsymbol{f}$ is available, it can be used to inform the mean function $\boldsymbol{f}$ (potentially also covariance $\boldsymbol{\Sigma}_D$) appropriately. In particular, if the parametric form of the dynamics is known, we can use them as a direct substitute for $\boldsymbol{f}$. This is the special case of ODE-parameter inference, which we investigate empirically in Appendix A.3.4. Next we present implementation details of both non-parametric (i.e. $\boldsymbol{f}$ given by a neural network) and parametric (i.e. $\boldsymbol{f}$ given by some parametric form) dynamics models.

NON-PARAMETRIC DYNAMICS In the non-parametric case, we model both the dynamics' mean function $\boldsymbol{\mu}_D$ and covariance function $\boldsymbol{\Sigma}_D$ with a neural networks. Across all experiments, we choose a simple 3-layer neural network with 20, 20 and $2n$ nodes, where n denotes the number of state dimensions of a specific system. After each layer, we apply the sigmoid activation function, except for the last one. The first n output nodes represent the mean function. Here, no activation function is necessary for the last layer. The second n output nodes are used to construct the diagonal covariance $\boldsymbol{\Sigma}_D$. To ensure positivity, we use $x \mapsto \log(1 + \exp(x))^2$ as an activation function on the last n nodes of the last layer.

PARAMETRIC DYNAMICS In the parametric case, we model $\boldsymbol{\mu}_D$ using the parametric form of the vector field. Across all experiments, we choose a simple 3-layer neural network with 10, 10 and n nodes, where n denotes the number of state dimensions of a specific system. After each lyer, we apply the sigmoid activation function, except for the last one. The n nodes are then used to construct the diagonal covariance $\boldsymbol{\Sigma}_D$. To ensure positivity, we use $x \mapsto \log(1 + \exp(x))^2$ as an activation function on the last layer.

A.3.2.2 *Smoother model*

The role of the smoother model is to map every tuple $(\boldsymbol{x}(0), t)$ consisting of initial condition $\boldsymbol{x}(0)$ and time t to $\boldsymbol{x}(t)$, which is the state at time t of a trajectory starting at $\boldsymbol{x}(0)$ at time 0. In the paper, we model the smoother using a Gaussian process with a deep mean function $\boldsymbol{\mu}$ and a deep feature map $\boldsymbol{\phi}$. Both of them take as input the tuple $(\boldsymbol{x}(0), t)$. This tuple is then mapped through a dense neural network we call core. For all experiments, we chose a core with two layers, with 10 and 5 hidden nodes and sigmoid activation on both. The output of the core is then fed into two linear heads. The head for $\boldsymbol{\mu}$ builds a linear combination of the core's output to obtain a vector of the same shape as $\boldsymbol{x}(t)$. The head for $\boldsymbol{\phi}$ builds a linear combination of the core's output to obtain a vector of length 3, the so called features. These features are then used as inputs to a standard RBF kernel with ARD [Ras04b]. For each state dimension, we keep a separate $\boldsymbol{\phi}$-head, as well as separate kernel hyperparameters. However, the core is shared across dimensions, while $\boldsymbol{\mu}$ is directly introduced as multidimensional.

In the paper, we set the variance of the RBF to 1 and learned the lengthscales together with all other hyperparameters. However, due to the expressiveness of the neural network, the lengthscales are redundant and could easily be incorporated into the linear combination performed by the head. Thus, in the scaling experiments, we fix the lengthscale to one and approximate the RBF kernel with a feature expansion, as detailed in Section 4.4.

A.3.2.3 *Evaluation metric*

To evaluate the quality of our models' predictions, we use the log likelihood. To obtain the log likelihood, we first use the model to predict the mean and standard deviation at 100 equidistant times. Then we calculate the log likelihood of the ground truth for every predicted point. We take the mean over dimensions, over times, and over trajectories. When reporting the training log likelihood, as done e.g. in Table 4.1, we use the training trajectories for evaluation. When reporting the generalization log likelihood, as done e.g. in Table 4.2, we use 10 unseen trajectories. This evaluation is then repeated for 10 different , meaning that we retrain the model 10 times on a data set with the same ground truth, but a different noise realization. We then report the mean and standard deviation of the log likelihood across these repetitions.

WEIGHT DECAY To prevent overfitting, we use weight decay on the parameters of both the dynamics and the smoother neural networks. We denote by wd_D the weight decay parameter of the dynamics model, and wd_S the weight decay parameters of the smoother model. While we keep the wd_S constant during all three phases of training, we gradually increase wd_D from 0 to its final value, which is the same as wd_S. The increase follows a polynomial schedule with power 0.8.

A.3.2.4 *Training details*

The training of DGM, i.e. optimizing Equation (4.3), can be split into three distinct phases: *transition*, *training*, and *fine-tuning*. In the *transition* phase we gradually increase the value of both λ and the weight decay regularization parameter of the dynamics (wd_D) from 0 to its final value. When these parameters reach their final value, we reach the end of the transition phase and start the *training* phase. In this phase, all optimization parameters are left constant. It stops when the last 1000 steps are reached. Then, the *fine-tune* phase starts, where we decrease learning rate to 0.01. The gradual increase of λ and wd_D follows polynomial schedule with power 0.8. As an optimizer, we use Adam.

SUPPORTING POINTS The selection of the supporting points in $\mathcal{T}$ is different for data sets consisting of one or multiple trajectories. If there is only one trajectory in the dataset, we match the derivatives at the same places where we observe the state. If there are multiple trajectories, we match the derivatives at 30 equidistant time points on each training trajectory.

SELECTION OF λ The loss of Equation (4.3) is a multi-objective optimization problem with a trade-off parameter λ. Intuitively, if λ is too small, the model only tries to fit the data and neglects the dynamics. On the other hand, with too large λ, the model neglects the data fit and only cares about the dynamics. In Figure A.43 we show a plot of log likelihood score on the 10 test trajectories of the LV 100 dataset with varying λ. We train the model for $\lambda \cdot |\dot{\mathcal{X}}|/|\mathcal{D}| \in \{2^i | i = -20, \dots, 6\}$. To estimate the robustness of the experiment, we show the mean and standard deviation over 5 different noise realizations.

PARAMETER SELECTION To search for the best performing parameters we performed sweep over learning rate value lr in the transition and training

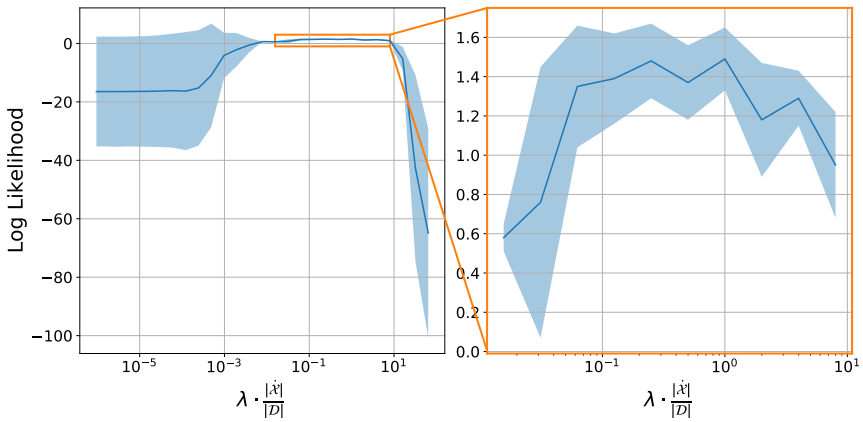


FIGURE A.43: If λ is too small, the dynamics model does not regularize the smoother sufficiently, the model overfits to the data and the test log likelihood score is worse. If λ is too large, the observation term gets dominated, the model underfits and the log likelihood score on the test data is worse. Empirically, we found that we achieve the best log likelihood on the test data with $\lambda = |\mathcal{D}|/|\dot{\mathcal{X}}|$.

phase and over the weight decay parameters wd_S. For lr, we considered the values $0.02, 0.05$ and 0.1. For wd_S, we considered $0.1, 0.5$ and 1.0.

TRAINING TIME We did not optimize our hyperparameters and code for training time. Nevertheless, we report the length of each phase and the total time in the Table A.1.

	Transition	Training	Fine-Tuning	Time[s]
LV 1	1000	0	1000	329 ± 15
LO 1	1000	0	1000	399 ± 6
DP 1	1000	1000	1000	535 ± 42
QU 1	1000	2000	1000	1121 ± 41
LV 100	1000	0	1000	408 ± 8
LO 125	1000	0	1000	753 ± 8
DP 100	5000	4000	1000	2988 ± 261
QU 64	6000	3000	1000	8387 ± 36

TABLE A.1: Number of steps for each training phase and total training time for different datasets. For the times, we report mean $\pm$ standard deviation over 10 different random seeds.

A.3.3 *Bayesian NODE training*

In this section, we describe the specifics of the experiments and the implementation of SGLD and SGHMC, the Bayesian integration benchmarks used in the paper. For all experiments, we use a slightly changed version of the code provided by Dandekar et al. [Dan+21], which is written in Julia [Bez+17].

A.3.3.1 *Effects of Overparametrization*

Here we provide further details regarding the experiment presented in Figure 4.2. For the ground truth dynamics $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$, we selected the matrix $\boldsymbol{A}$ such that it has 1 stable and 2 marginally stable modes. The eigenvalue of the stable mode is selected uniformly at random from $[-0.5, -0.1]$. To create marginally stable modes, we create a block $\boldsymbol{C} \in \mathbb{R}^{2\times 2}$, where its components are sampled i.i.d. uniformly at random from $[0, 1]$. The marginally stable part is then created as $\boldsymbol{A}$ as $\frac{\pi}{2\rho(\boldsymbol{C}-\boldsymbol{C}^\top)}\left(\boldsymbol{C}-\boldsymbol{C}^\top\right)$, where $\rho(.)$ denotes the spectral radius. Using the spectral radius in the normalization ensures that the period of the mariginally stable mode is bounded with $\pi/2$. We selected the initial condition for the trajectory uniformly at random from the unit sphere in $\mathbb{R}^3$. We evolved the trajectory on the time interval $(0, 10)$ and observed 100 noisy observations, where every ground truth value

was perturbed with additive, independent Gaussian noise, drawn from $\mathcal{N}\left(0, 0.1^2\right)$.

While DGM performed without any pre-training, we observed that both SGLD and SGHMC struggle without narrow priors. To obtain reasonable priors, we followed the following procedure:

First, we pretrain every set of matrices $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_k$ on the ground truth, such that the

$$\sum_{i=1}^{100} \left\| \dot{\boldsymbol{x}}(t_i) - \prod_{j=1}^{k} \boldsymbol{B}_j \boldsymbol{x}(t_i) \right\|_2^2 \leq 10^{-5},$$

where t_i are times at which we observed the state. We selected the prior as Gaussian centered around the pretrained parameters. The standard deviation was chosen such that the standard deviation of the product of the matrices $\prod_{j=1}^{k} \boldsymbol{B}_j$ stays at 0.01, independent of the number of matrices used. For SGHMC, we selected learning rate 1.5×10^{-7} and momentum decay 0.1 as hyperparameters. For SGLD,we chose the hyperparameters $a = 0.001, b = 1$ and $\gamma = 0.9$, which SGLD then uses to calculate a polynomial decay $a(b+k)^{-\gamma}$ (at step k) for its learning rate. For sampling we used 5 chains with 20000 samples on each chain. The last 2000 samples of each chain were used for evaluation. With this setting we ensured that the r-hat score was smaller than 1.1.

A.3.3.2 *Finetuning for Benchmark Systems*

Both SGLD and SGHMC require hyperparameters, that influence their performance quite strongly. In this subsection, we explain all the tuning steps and requirements we deployed for these algorithms to produce the results shown in the main paper. For both algorithms, we used a setup of 6 chains, that were sampled in parallel, with 10000 samples per chain, where the final 2000 samples were taken as predictions. These numbers were selected using the r-hat value to determine if the chains had sufficiently converged.

HYPERPARAMETERS OF SGLD For SGLD, we additionally searched over the hyperparameters a, b, and γ. All three are used to calculate the learning rate of the algorithm. These parameters were chosen by evaluating the log likelihood of the ground truth on a grid, which was given by the values $a \in \{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1\}$, $b \in \{0.3, 0.6, 1.0, 1.5, 2\}$ and $\gamma \in \{0.5001, 0.55, 0.6, 0.7, 0.8, 0.99\}$. Clearly, using the log likelihood of the

ground truth to tune the hyperparameters overestimates the performance of the algorithm, but it provides us with an optimistic estimate of its real performance in practice. For computational reasons, we only performed a grid search on the one trajectory experiments, and then reused the same hyperparameters on the multi-trajectory experiments. All hyperparameters are shown in Table A.2.

HYPERPARAMETERS OF SGHMC For SGHMC, we additionally searched over the hyperparameters the learning rate and the momentum decay. Again, these parameters were chosen by evaluating the log likelihood of the ground truth on a grid, where learning rate was chosen from the set $\{1e{-}8, 5e{-}8, 1.5e{-}7, 5e{-}7, 1e{-}6, 5e{-}6, 1e{-}5, 5e{-}5, 1e{-}4, 5e{-}4\}$ and momentum decay was chosen from the set $\{0.0001, 0.001, 0.05, 0.1, 0.5, 1, 5\}$. Since we used the log likelihood of the ground truth again to tune the hyperparameters, we overestimate the performance of the algorithm and obtain thus an optimistic estimate of its real performance in practice. For computational reasons, be only performed a grid search on the one trajectory experiments, and then reused the same hyperparameters on the multi-trajectory experiments. All hyperparameters are shown in Table A.2.

	SGLD			SGHMC	
	a	b	γ	learning rate	momentum decay
LV p	1e−3	2	0.5001	5e−7	0.1
Lo p	0.001	1.5	0.5001	1e−5	0.05
DP p	0.1	0.3	0.5001	1e−6	0.1
QU p	0.0001	1.5	0.7	5e−7	0.5
LV n	0.005	1.5	0.7	5e−7	0.5
LO n	0.001	1.5	0.55	5e−6	0.1
DPn	0.01	2	0.55	1e−6	0.05
QU n	F	F	F	5e−7	0.05

TABLE A.2: Hyperparameters with best performance evaluated on the likelihood of the ground truth. The hyperparameters are different for the parametric (p) and the non-parametric (n) dynamics models, which is indicated with the last letter.

CHOICE OF PRIORS FOR SGLD AND SGHMC Since SGLD and SGHMC are both Bayesian methods, they need to be supplied with a prior. As we discovered in our experiments, this prior plays a crucial role in the stability of the algorithm. In the end, we did not manage to get them to converge without some use of ground truth. In particular, if the priors were not chosen narrowly around some ground truth, the algorithms just returned NaNs, since their integration scheme runs into numerical issues. For the parametric case shown in Appendix A.3.4, where we assume access to the true parametric form of the system, we thus chose narrow priors around the ground truth of the parameter values, that were used to create the data set. For Lotka Volterra, we chose a uniform distribution around the ground truth ± 0.5, i.e. $\theta_i \sim \text{Uniform}[0.5, 1]$ for all components of $\boldsymbol{\theta}$. For Lorenz, we chose $\alpha \sim \text{Uniform}[8, 12]$, $\beta \sim \text{Uniform}[25, 31]$ and $\gamma \sim \text{Uniform}[6/3, 10/3]$. For Double Pendulum, we chose $m \sim \text{Uniform}[0.5, 1.5]$ and $l \sim \text{Uniform}[0.5, 1.5]$. For Quadrocopter, we chose independent Gaussians, centered around the ground truth, with a standard deviation of 0.005. For all experiments, the prior on the observation noise was set to $\sigma \sim \text{InverseGamma}[2, 3]$, except for SGLD when inferring the Lorenz system. There, we had to fix the noise standard deviation to its ground truth, to get convergence.

For the non-parametric case shown in the main paper, we needed a different strategy, since no ground truth information was available for the weights of the neural dynamics model. Thus, we first trained a deterministic dynamics model. As data, we sampled 100 tuples $\boldsymbol{x}, \dot{\boldsymbol{x}}$ equidistantly in time on the trajectory. Note that we implicitly assume access to the ground truth of the dynamics model, i.e. we assume we are provided with accurate, noise free $\dot{\boldsymbol{x}}$. The neural dynamics model was then pre-trained on these pairs, until the loss was almost zero (up to $1e{-}5$). SGLD and SGHMC were then provided with Gaussian priors, independent for each component of $\boldsymbol{\theta}$, centered around the pre-trained weights, with a standard deviation of 0.1.

A.3.3.3 *Number of integrations for prediction*

SGLD and SGHMC both return samples of the parameters of the dynamics model. To obtain uncertainties in the state space at prediction time, each one of these samples needs to be turned into a sample trajectory, by using numerical integration. To obtain maximum accuracy, we would ideally integrate all parameter samples obtained by the chains. However, due to the computational burden inflicted by numerical integration, this is not feasible.

We thus need to find a trade-off between accuracy and computational cost, by randomly subsampling the number of available parameter samples.

In Figure A.44 we show how the log likelihood of the ground truth changes with increasing number of sample trajectories on the LV 1 dataset. After initial fluctuations, the log likelihood of the ground truth stabilizes after approximately 200 steps. To obtain the results of Table 4.1, we thus chose 200 integration steps.

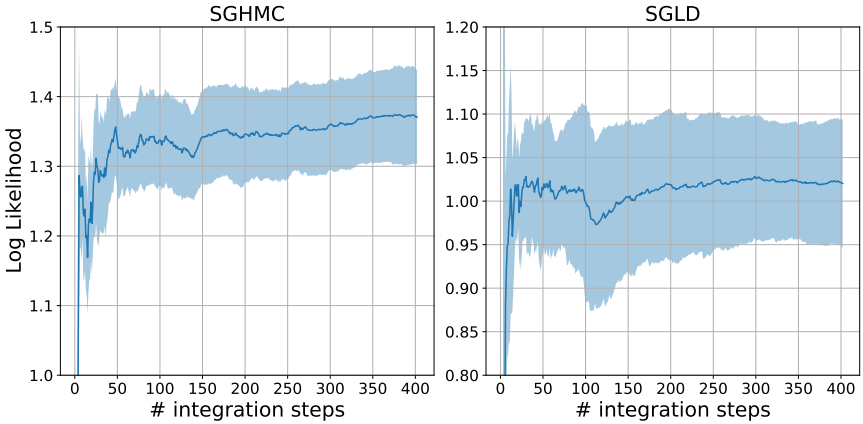


FIGURE A.44: We select the number of sample trajectories for uncertainty prediction to be 200, since we observe that the log likelihood of the ground truth stops fluctuating after 200 steps.

A.3.4 *Additional experiments*

In this section, we first show the state predictions of DGM on the datasets with multiple trajectories. Then, we compare DGM with SGLD and SGHMC for the parametric case, i.e. where we assume to have access to the true parametric form of the dynamics. Since most datasets have too many trajectories to be fully visualized, we show a random subset instead.

A.3.4.1 *Sample plots from trained trajectories*

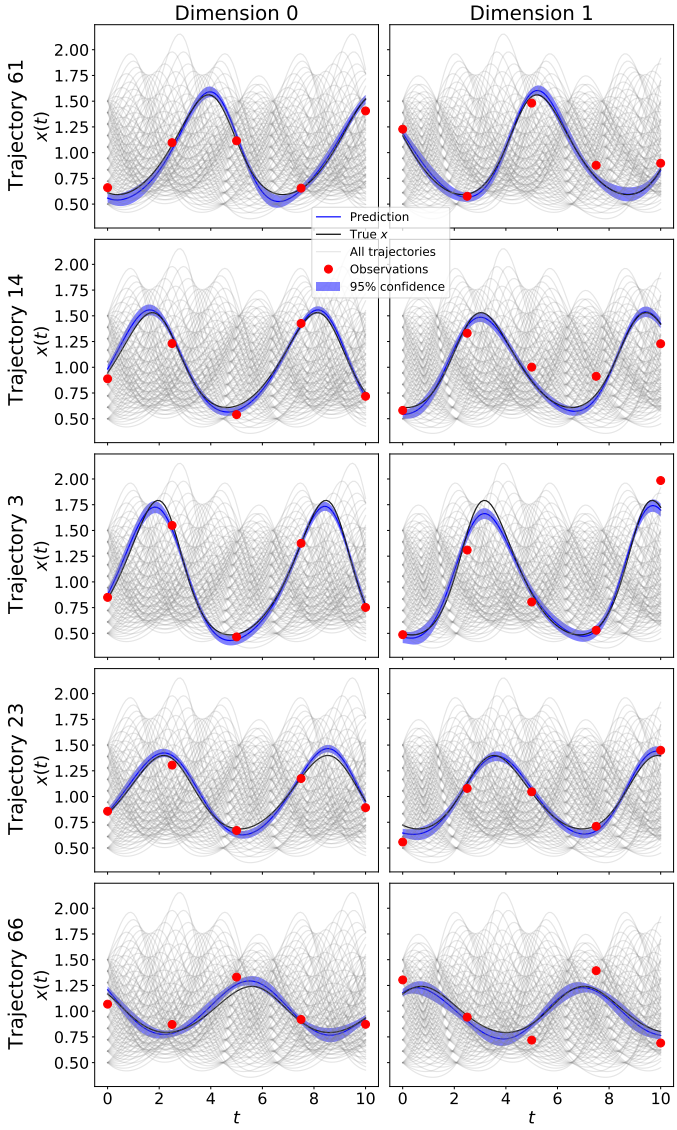


FIGURE A.45: DGM's prediction on 5 randomly sampled training trajectories of LV 100.

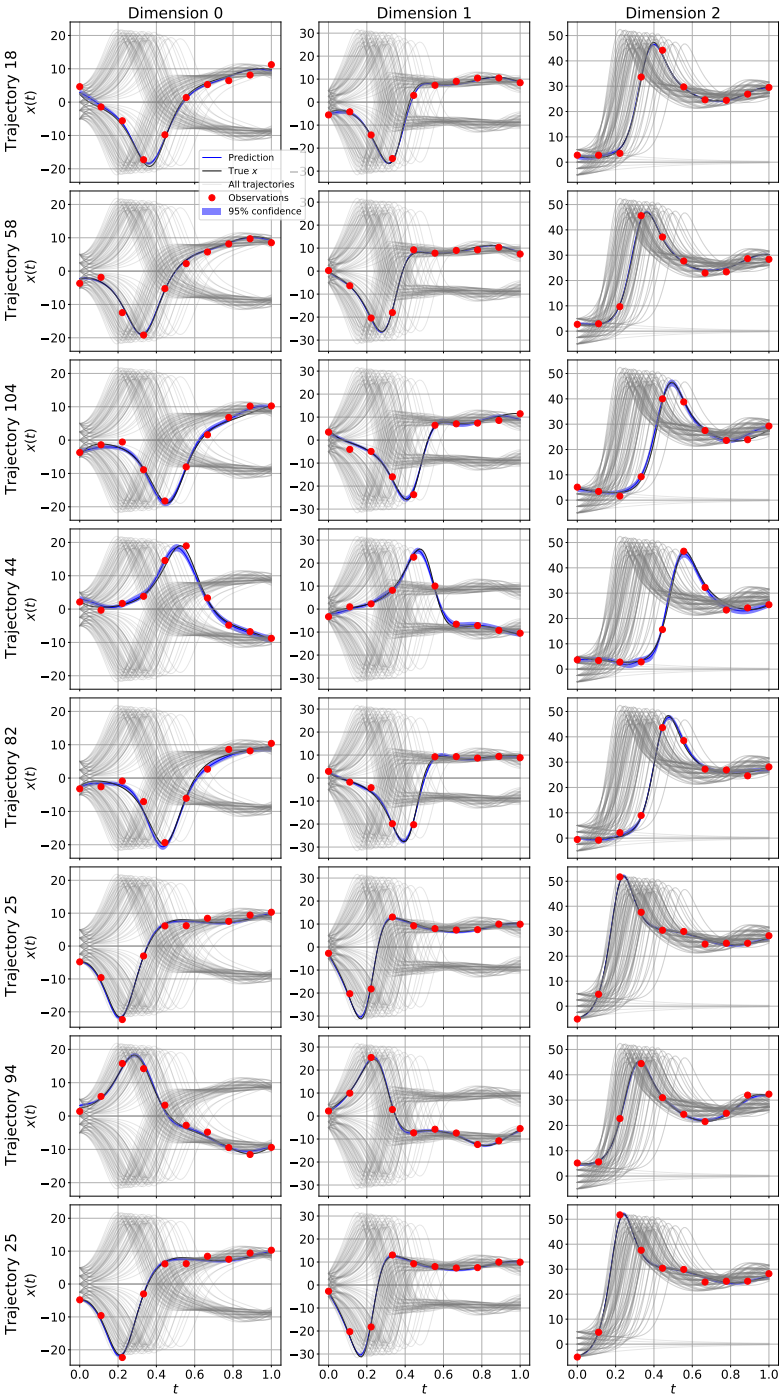


FIGURE A.46: DGM's prediction on 8 randomly sampled training trajectories of

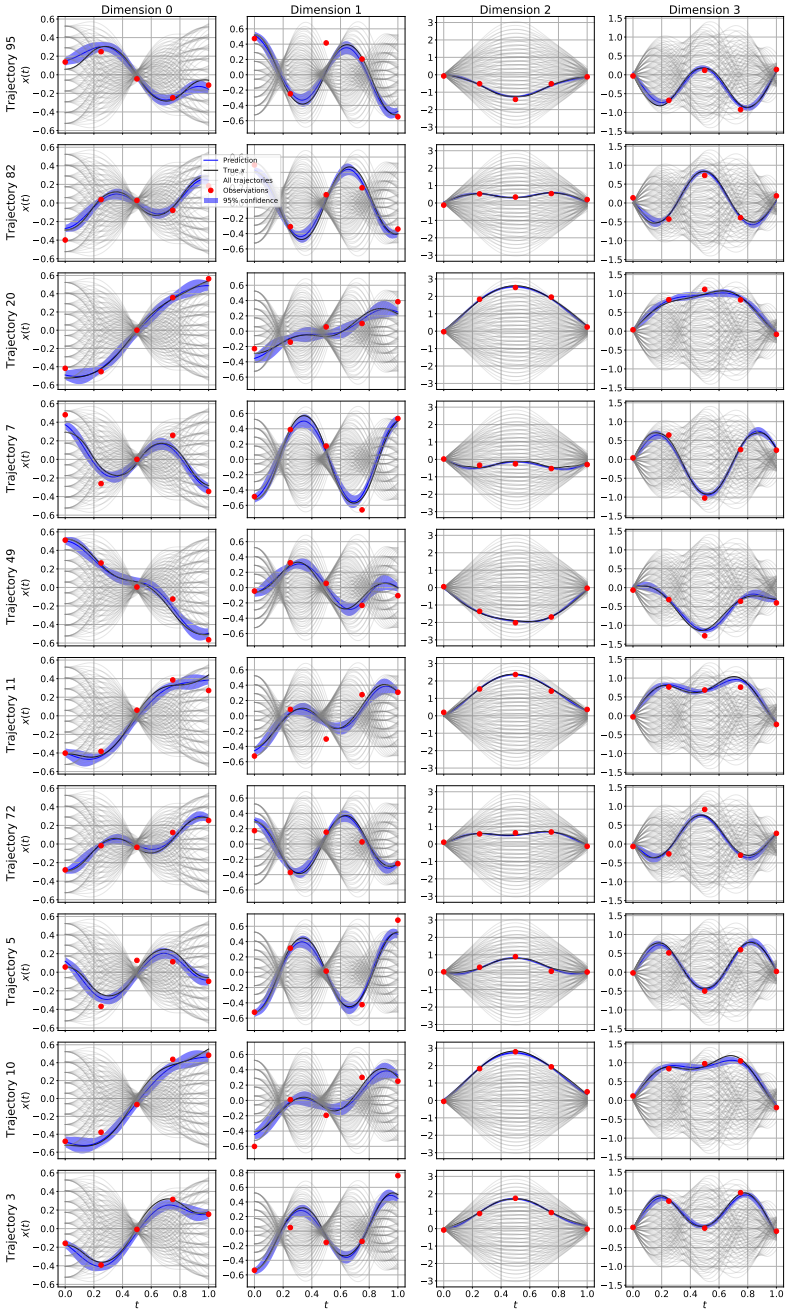


FIGURE A.47: DGM's prediction on 10 randomly sampled training trajectories of DP 100.

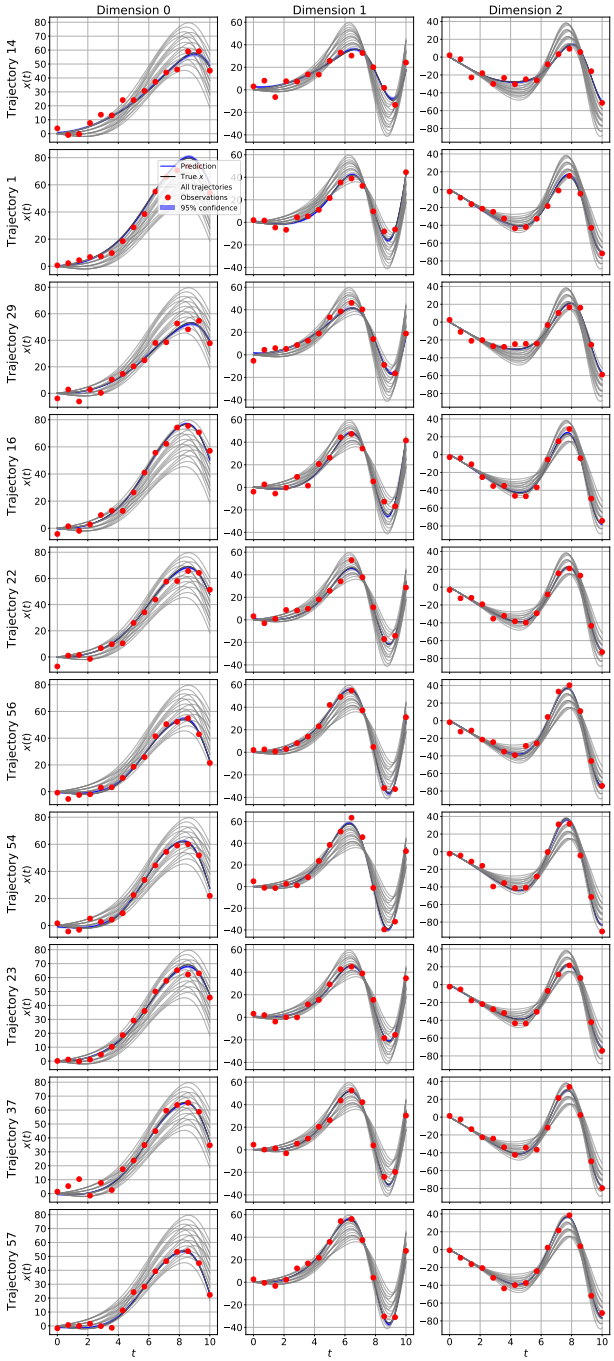


FIGURE A.48: DGM's prediction on 10 randomly sampled training trajectories of OU 64, for state dimensions 0-2.

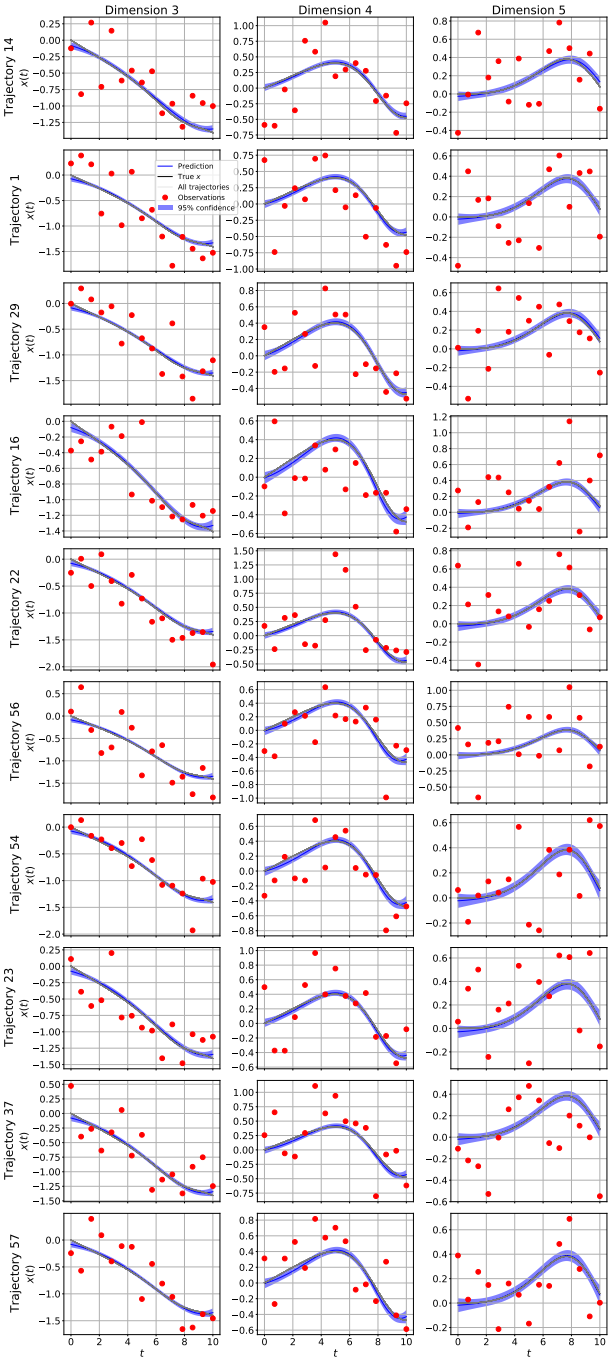


FIGURE A.49: DGM's prediction on 10 randomly sampled training trajectories of OU 64, for state dimensions 3-5.

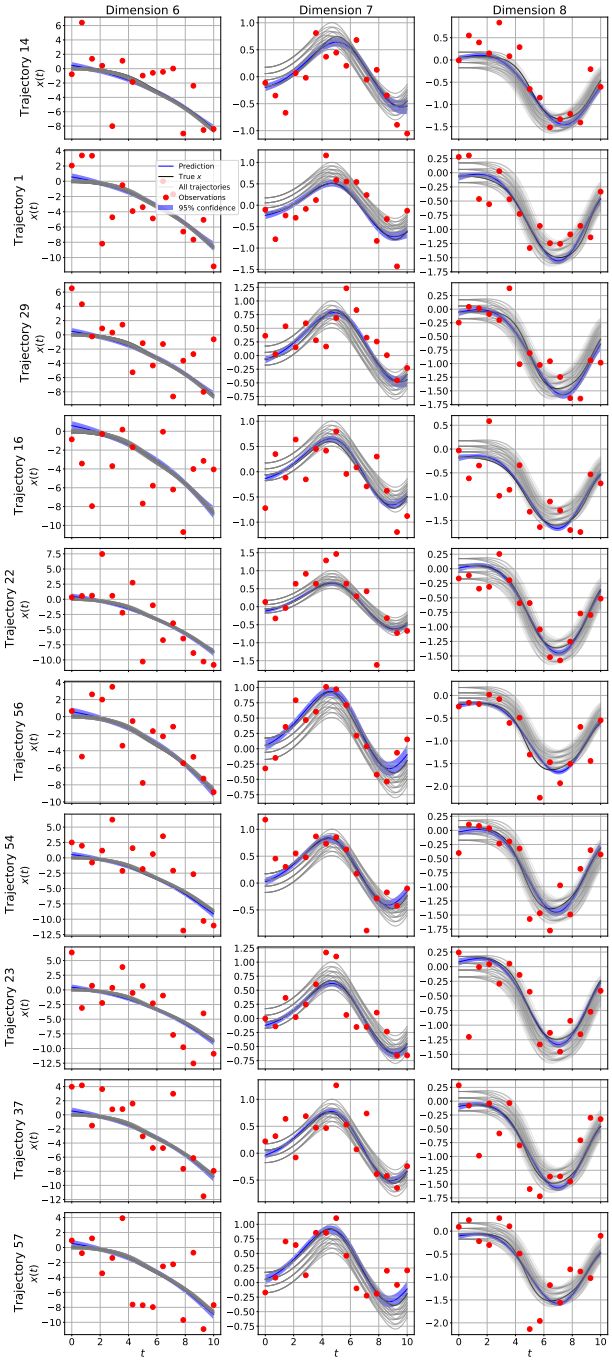


FIGURE A.50: DGM's prediction on 10 randomly sampled training trajectories of OU 64, for state dimensions 6-8.

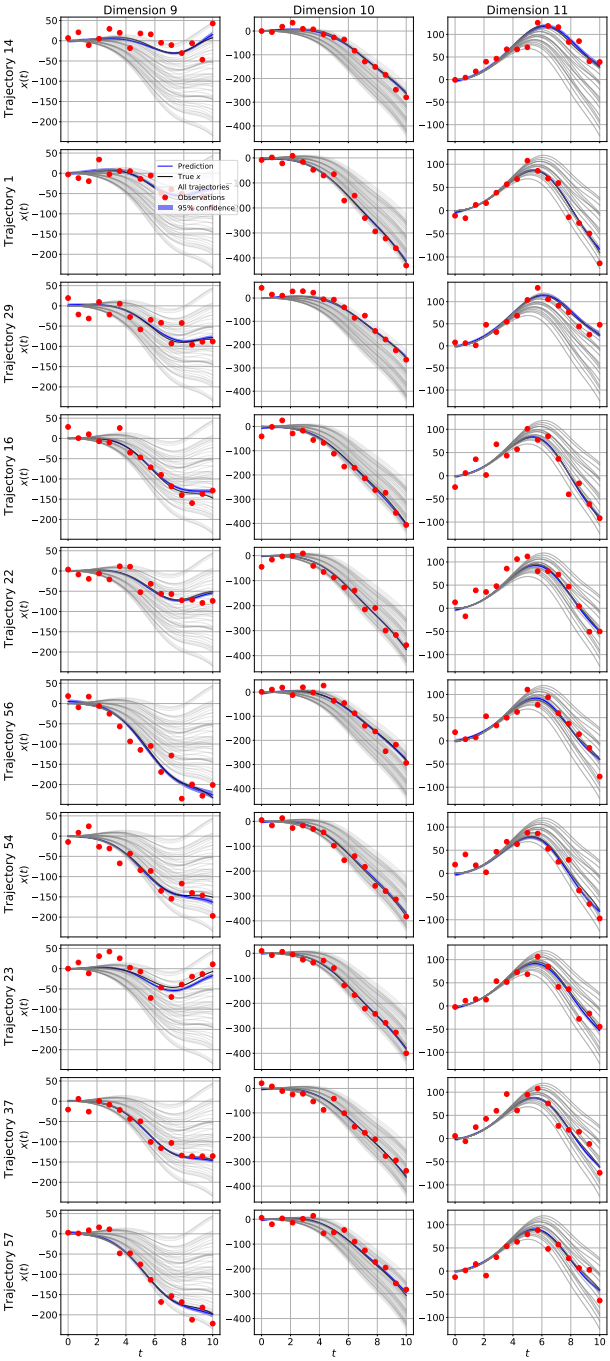


FIGURE A.51: DGM's prediction on 10 randomly sampled training trajectories of OU 64, for state dimensions 9-11.

A.3.4.2 *Sample plots from test trajectories*

Here, we show DGM's predictions on the test trajectories used to test generalization, as introduced in Appendix A.3.1. Since LV 100 is a two dimensional system, we also show the placement of the train and test initial conditions in Figure A.52.

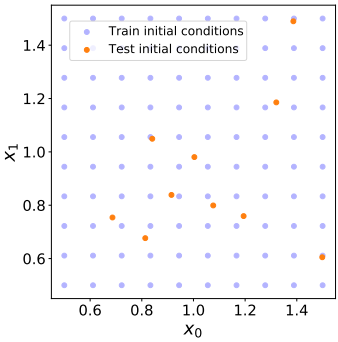


FIGURE A.52: Placement of the initial conditions for the train and test trajectories of the LV 100 dataset. We selected the initial conditions for the train trajectories by gridding $[0.5, 1.5]^2$ with 10 points in every dimension. We select initial conditions for test trajectories independently, uniformly at random from the cube $[0.5, 1.5]^2$.

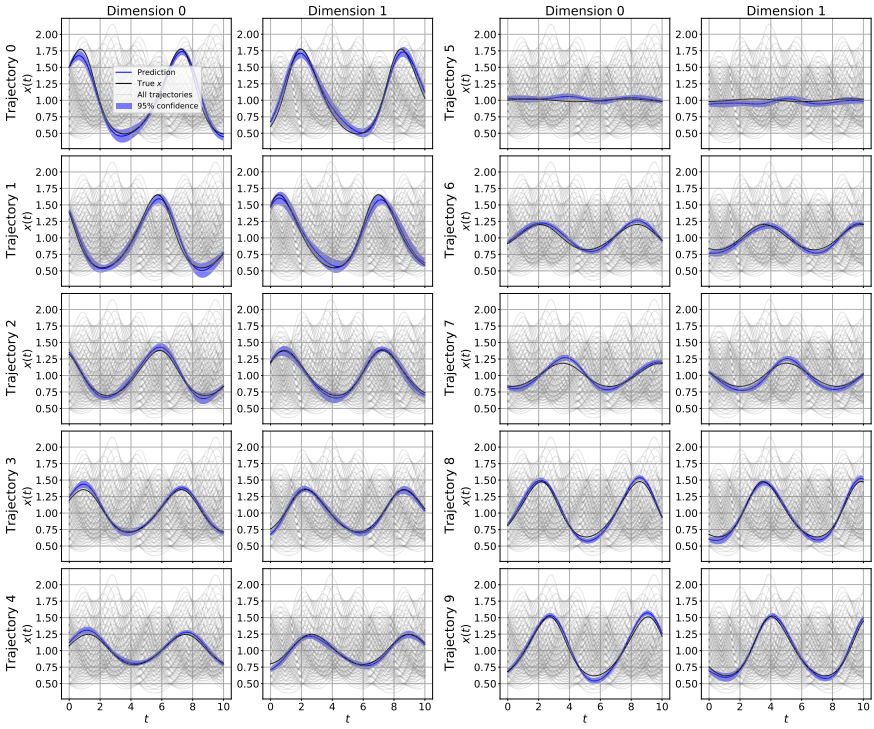


FIGURE A.53: DGM's prediction on 10 randomly sampled test trajectories for the LV 100 dataset.

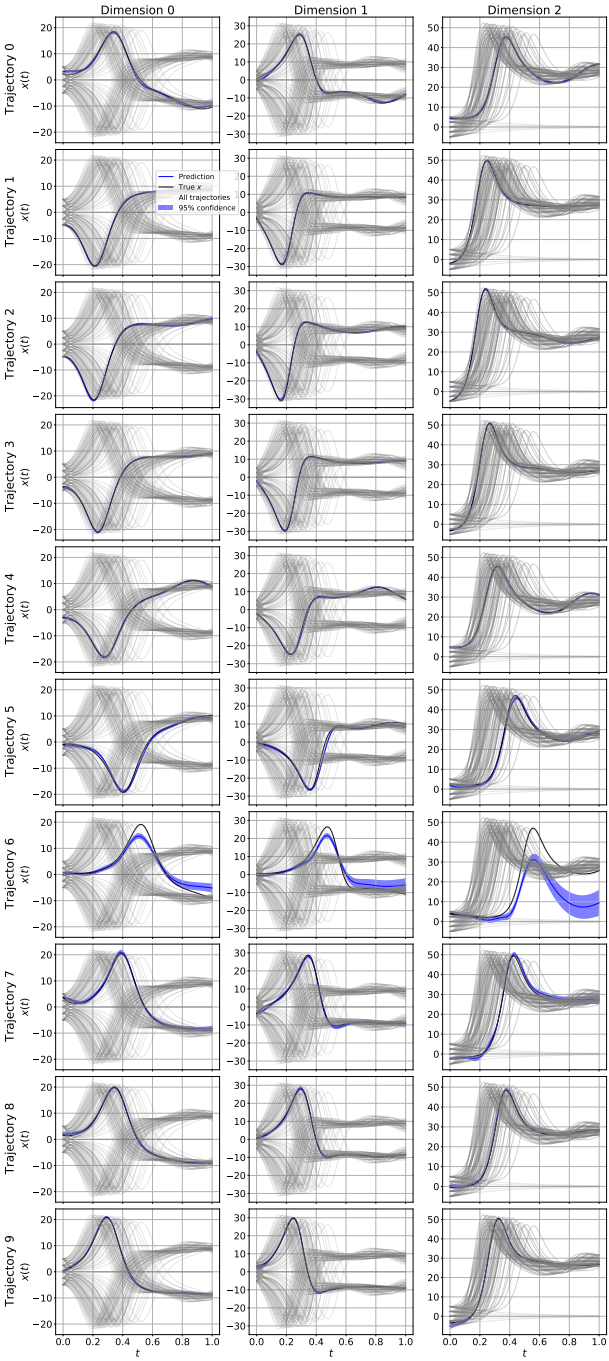


FIGURE A.54: DGM's prediction on 10 randomly sampled test trajectories for the LO 125 dataset.

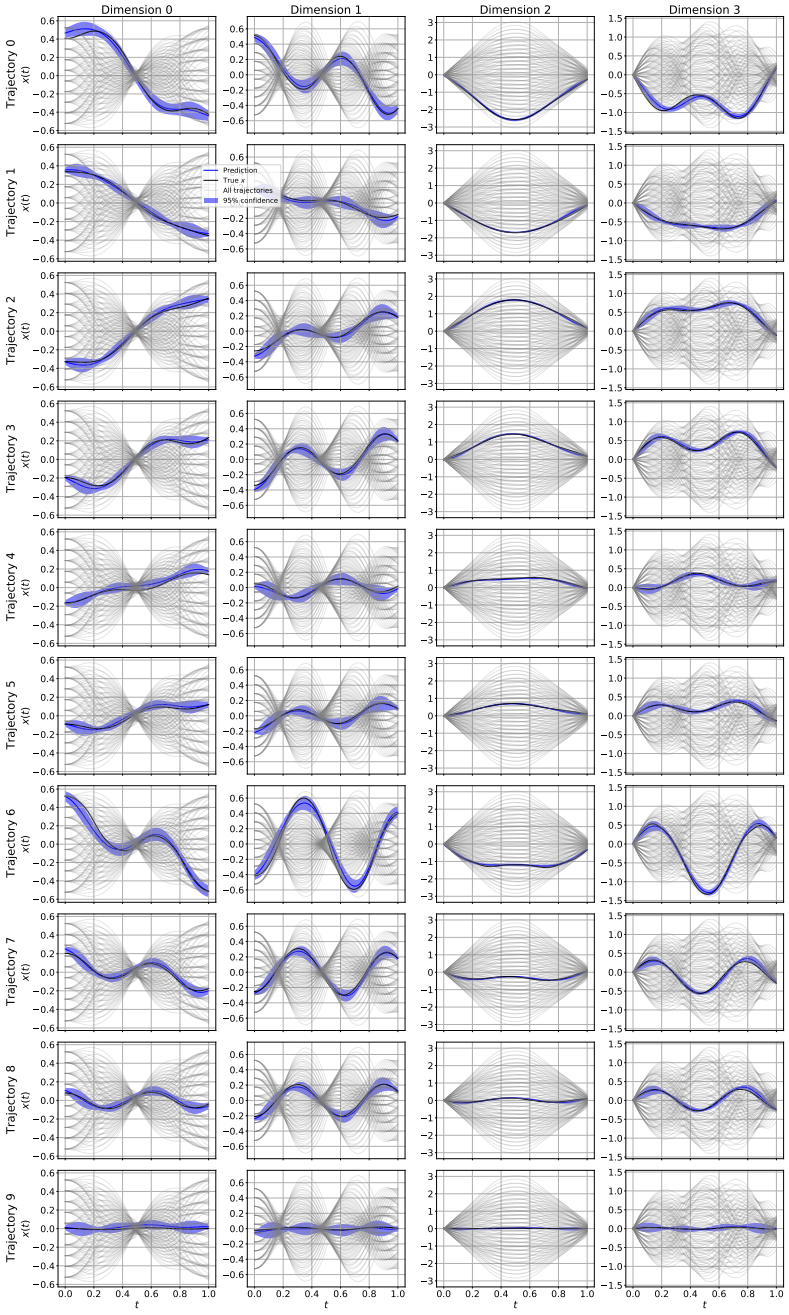


FIGURE A.55: DGM's prediction on 10 randomly sampled test trajectories for the DP 100 dataset.

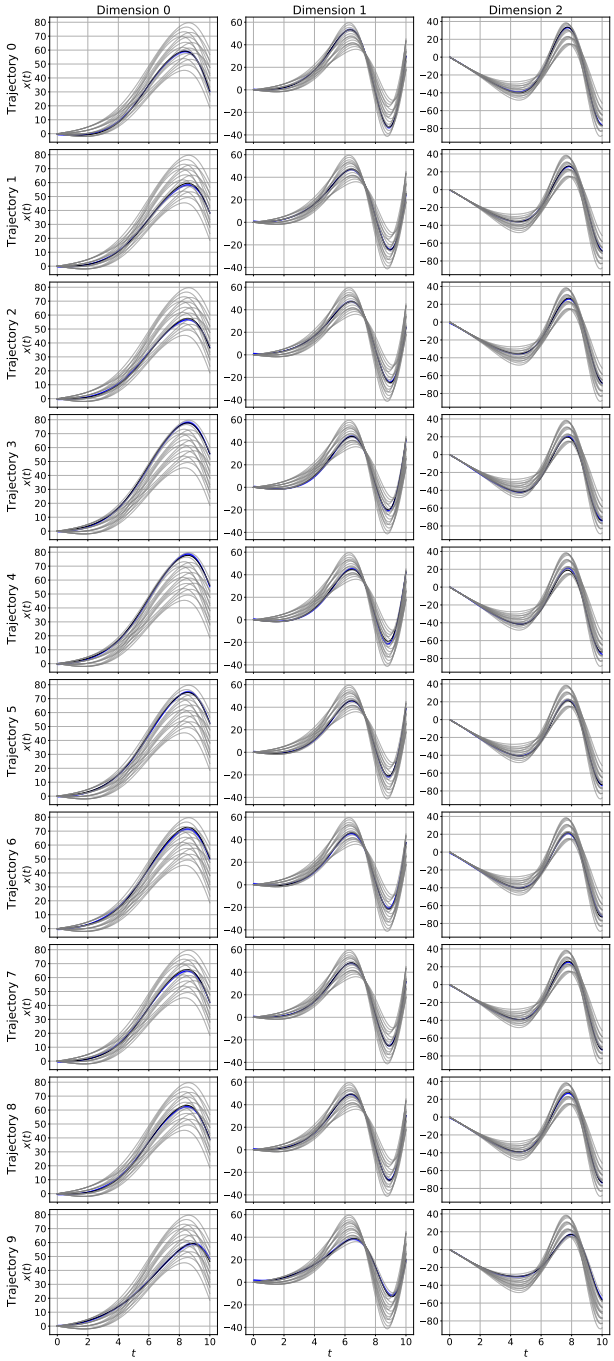


FIGURE A.56: DGM's prediction on 10 randomly sampled test trajectories of QU 64, for state dimensions 0-2.

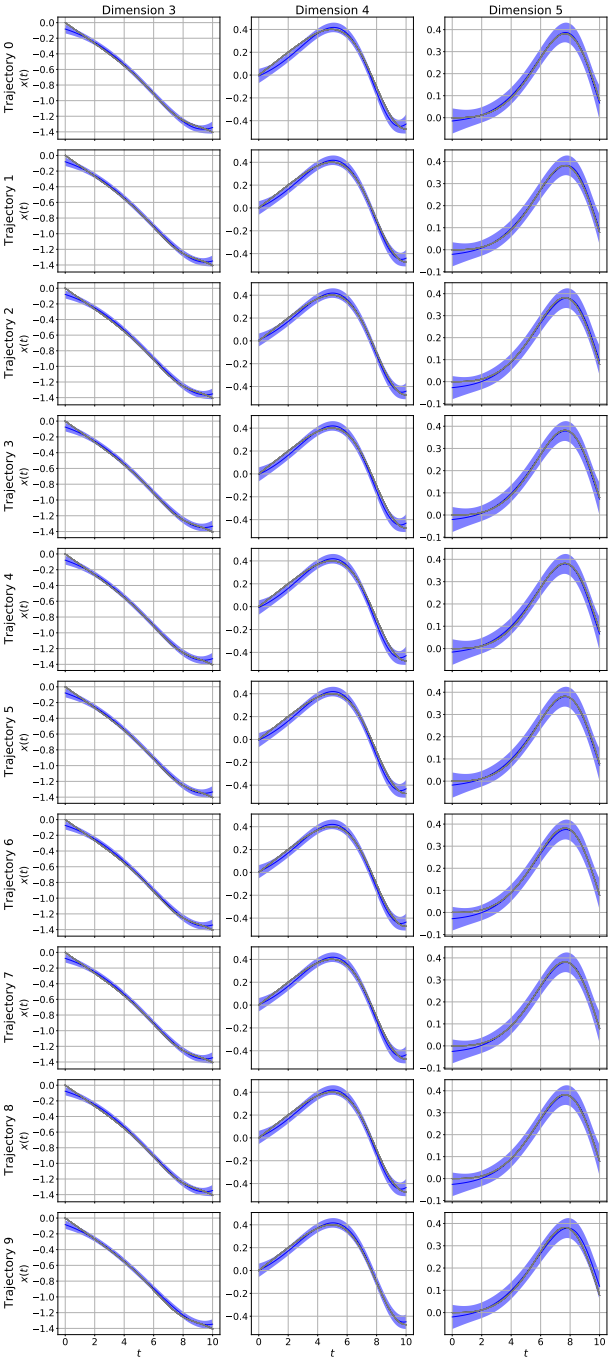


FIGURE A.57: DGM's prediction on 10 randomly sampled test trajectories of QU 64, for state dimensions 3-5.

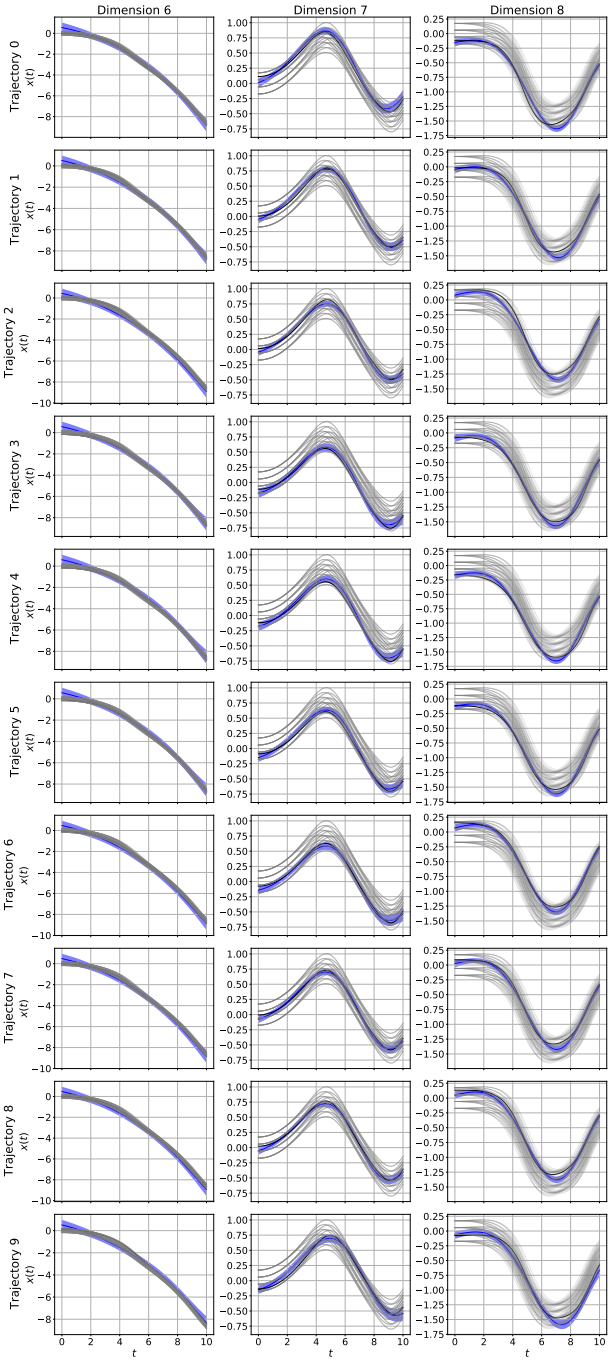


FIGURE A.58: DGM's prediction on 10 randomly sampled test trajectories of QU 64, for state dimensions 6-8.

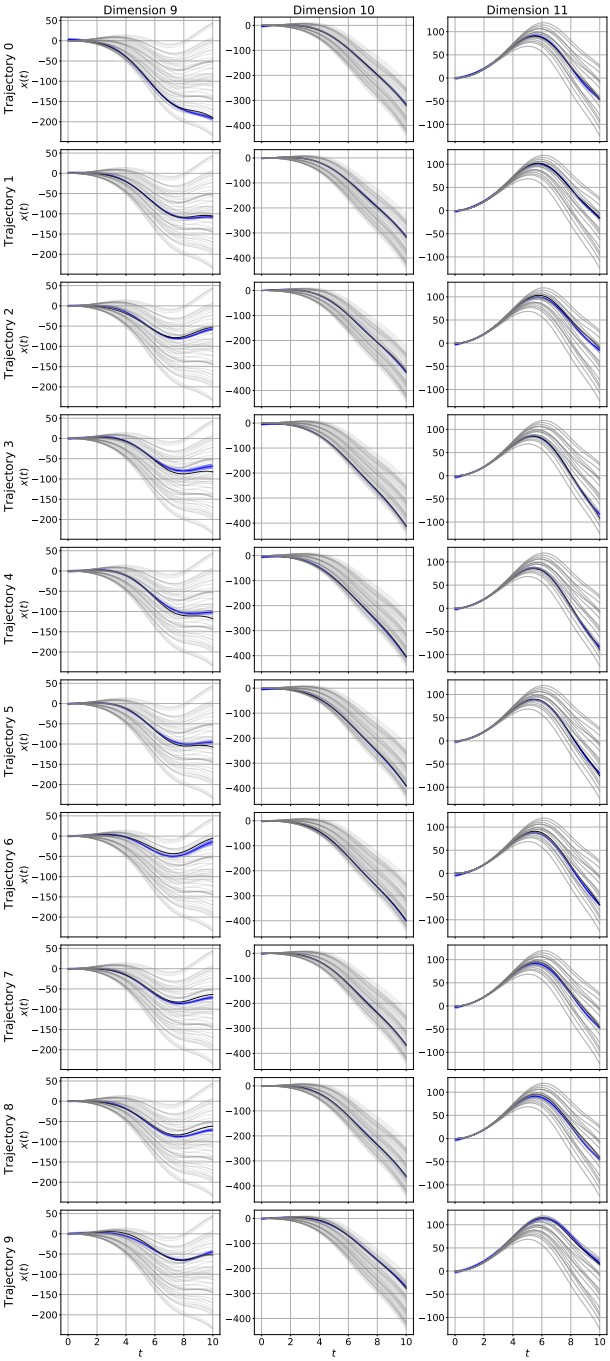


FIGURE A.59: DGM's prediction on 10 randomly sampled test trajectories of QU 64, for state dimensions 9-11.

A.3.4.3 *Comparison with parameteric integration*

In this subsection, we compare DGM against SGLD and SGHMC in the parametric setting, i.e. where we assume access to the parametric form of the true dynamics $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$. Despite serious tuning efforts outlined in Section refsubsection: DGM hyperparameters selection, we were unable to make SGLD and SGHMC perform on any multitrajectory experiments except for Lotka Volterra 100. As can be seen in Table A.3, the sampling based methods seem to perform quite well. However, it should be noted that we were unable to get a stable performance without using ground truth information, as outlined in Section A.3.3.2. Given this caveat and the results of the non-parametric case in the main paper, we conclude the following. If strong and accurate expert knowledge is available that can be used to fix strong priors on simple systems, the sampling-based approaches are certainly a good choice. For more complex systems or in the absence of any expert knowledge, DGM seems to have a clear edge.

	Log Likelihood		
	DGM	SGLD	SGHMC
LV 1	1.98 ± 0.18	$\mathbf{3.07 \pm 0.685}$	3.06 ± 0.517
LO 1	-0.52 ± 0.09	$\mathbf{2.01 \pm 0.548}$	F
DP 1	2.16 ± 0.13	$\mathbf{3.43 \pm 0.396}$	2.96 ± 0.795
QU 1	0.71 ± 0.07	$\mathbf{2.42 \pm 0.322}$	1.38 ± 0.00
LV 100	1.85 ± 0.11	$\mathbf{4.28 \pm 0.184}$	4.26 ± 0.178

TABLE A.3: Log likelihood of the ground truth of 100 points on the training trajectories. SGHMC and SGLD were provided with strong, ground-truth-inspired priors and received an extensive hyperparameter sweep using the ground truth as metric. Nevertheless, DGM performs decently in comparison, without using neither priors nor ground truth.

A.4 APPENDIX TO ARES/MARS

A.4.1 *Parameter Estimation Lorenz '63*

Ground truth	NPSDE	ESGF	AReS	MaRS
$\theta_0 = 10$	1.28 ± 2.32	$\mathbf{9.97 \pm 0.33}$	7.24 ± 1.08	9.82 ± 0.56
$\theta_1 = 28$	20.69 ± 5.73	$\mathbf{28.00 \pm 0.17}$	28.16 ± 1.08	27.96 ± 0.21
$\theta_0 = 2.667$	1.86 ± 1.08	$\mathbf{2.65 \pm 0.06}$	2.55 ± 0.10	2.64 ± 0.07
$G = \sqrt{10}$	6.51 ± 1.31	$\mathbf{3.03 \pm 0.2}$	3.54 ± 2.45	

TABLE A.4: Median and standard deviation of the 65 best runs of each algorithm. As ESGF crashed in roughly one third of all experiments, we compare only the best 65 runs, where a crash is treated as a complete failure. While this provides somehow a fair comparison, it should be noted that this significantly overestimates the performance of all algorithms.

A.4.2 *Training Times*

	NPSDE	VGPA	ESGF	AReS	MaRS
OU Process	48.8 ± 0.9	$\sim (54 \pm 8)$hours	32.2 ± 0.3	321.3 ± 0.8	17.3 ± 0.3
DW Potential	406.9 ± 147.9	$\sim (12 \pm 6)$hours	35.2 ± 0.1	326.0 ± 1.8	17.7 ± 2.0
Lotka-Volterra	1421.8 ± 1.0	/	244.7 ± 1.2	47.6 ± 1.3	19.3 ± 1.1
Lorenz '63	39273.5 ± 8.9	/	670.7 ± 10.7	26274.0 ± 2529.8	721.1 ± 10.7

TABLE A.5: Computational times (in seconds) required for training the different algorithms.

A.4.3 *Densities for Ancestral Sampling of the SDE-Based Model*

Given the graphical model in Figure 5.1a, it is straightforward to compute the densities used in the ancestral sampling scheme in Algorithm 3. After marginalizing out $\dot{\boldsymbol{z}}$, the joint density described by the graphical model can be written as

$$p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{y} | \boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = p(\boldsymbol{o}|\boldsymbol{G}) p(\boldsymbol{z}|\boldsymbol{\phi}) p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{o}, \boldsymbol{\sigma}) \tag{A.137}$$

Substituting the densities given by Equations (5.11), (5.12), (5.14) and (5.22) yields

$$p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \boldsymbol{B}\boldsymbol{\Omega}\boldsymbol{B}^T)\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}})\mathcal{N}(\boldsymbol{y}|\boldsymbol{z} + \boldsymbol{o}, \boldsymbol{T}). \quad \text{(A.138)}$$

Using a change of variables to simplify notation, we write

$$p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}})\mathcal{N}(\boldsymbol{y}|\boldsymbol{z} + \boldsymbol{o}, \boldsymbol{T}). \quad \text{(A.139)}$$

This equation is now subsequently modified by observing that the product of two Gaussian densities in the same random variable is again a Gaussian density:

$$\begin{aligned} p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) &= \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}})\mathcal{N}(\boldsymbol{y}|\boldsymbol{z} + \boldsymbol{o}, \boldsymbol{T}) \\ &= \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}})\mathcal{N}(\boldsymbol{z}|\boldsymbol{y} - \boldsymbol{o}, \boldsymbol{T}) \\ &= \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{y} - \boldsymbol{o}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})\mathcal{N}(\boldsymbol{z}|\boldsymbol{m}_z, \boldsymbol{C}_z) \\ &= \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{o}|\boldsymbol{y}, \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})\mathcal{N}(\boldsymbol{z}|\boldsymbol{m}_z, \boldsymbol{C}_z) \\ &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}} + \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})\mathcal{N}(\boldsymbol{o}|\boldsymbol{m}_o, \boldsymbol{C}_o)\mathcal{N}(\boldsymbol{z}|\boldsymbol{m}_z, \boldsymbol{C}_z) \end{aligned} \quad \text{(A.140)}$$

where

$$\boldsymbol{m}_z = \boldsymbol{C}_z(\boldsymbol{T}^{-1}(\boldsymbol{y} - \boldsymbol{o})) \quad \text{(A.141)}$$

$$\boldsymbol{C}_z = (\boldsymbol{C}_{\boldsymbol{\phi}}^{-1} + \boldsymbol{T}^{-1})^{-1} \quad \text{(A.142)}$$

$$\boldsymbol{m}_o = \boldsymbol{C}_o(\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}\boldsymbol{y} \quad \text{(A.143)}$$

$$\boldsymbol{C}_o = (\tilde{\boldsymbol{\Omega}}^{-1} + (\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1})^{-1} \quad \text{(A.144)}$$

This formula can be further refined with the Woodbury identity, i.e.

$$\begin{aligned} \boldsymbol{C}_z &= (\boldsymbol{C}_{\boldsymbol{\phi}}^{-1} + \boldsymbol{T}^{-1})^{-1} \\ &= \boldsymbol{C}_{\boldsymbol{\phi}} - \boldsymbol{C}_{\boldsymbol{\phi}}(\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}\boldsymbol{C}_{\boldsymbol{\phi}} \\ &= \boldsymbol{C}_{\boldsymbol{\phi}}(\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}\boldsymbol{T} \end{aligned} \quad \text{(A.145)}$$

which leads to

$$\boldsymbol{m}_z = \boldsymbol{C}_{\boldsymbol{\phi}}(\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}(\boldsymbol{y} - \boldsymbol{o}) \quad \text{(A.146)}$$

and

$$\begin{aligned} \boldsymbol{C}_o &= (\tilde{\boldsymbol{\Omega}}^{-1} + (\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1})^{-1} \\ &= \tilde{\boldsymbol{\Omega}} - \tilde{\boldsymbol{\Omega}}(\tilde{\boldsymbol{\Omega}} + \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}\tilde{\boldsymbol{\Omega}} \\ &= \tilde{\boldsymbol{\Omega}}(\tilde{\boldsymbol{\Omega}} + \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}(\boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T}) \end{aligned} \quad \text{(A.147)}$$

which leads to

$$\boldsymbol{m}_o = \tilde{\boldsymbol{\Omega}}(\tilde{\boldsymbol{\Omega}} + \boldsymbol{C}_{\boldsymbol{\phi}} + \boldsymbol{T})^{-1}\boldsymbol{y} \tag{A.148}$$

Since we observe $\boldsymbol{y}$, we are interested in calculating the conditional distribution

$$p(\boldsymbol{o}, \boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = \frac{p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma})}{p(\boldsymbol{y}|\boldsymbol{\phi}, \boldsymbol{G}.\boldsymbol{\sigma})} \tag{A.149}$$

Conveniently enough, the marginal density of $\boldsymbol{y}$ is already factorized out in Equation (A.140) (compare Equation (5.23)). Thus, we have

$$p(\boldsymbol{o}, \boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = \mathcal{N}(\boldsymbol{o}|\boldsymbol{m}_o, \boldsymbol{C}_o)\mathcal{N}(\boldsymbol{z}|\boldsymbol{m}_z, \boldsymbol{C}_z) \tag{A.150}$$

As $\mathcal{N}(\boldsymbol{o}|\boldsymbol{m}_o, \boldsymbol{C}_o)$ is independent of $\boldsymbol{z}$, we can employ ancestral sampling by first obtaining a sample of $\boldsymbol{o}$ through $\mathcal{N}(\boldsymbol{o}|\boldsymbol{m}_o, \boldsymbol{C}_o)$, and then utilizing such sample to get $\boldsymbol{z}$ through $\mathcal{N}(\boldsymbol{z}|\boldsymbol{m}_z, \boldsymbol{C}_z)$.

A.4.4 *Calculating the GP Posterior for Data-Based Ancestral Sampling*

Given the graphical model in Figure 5.1b, we can calculate the densities used in the ancestral sampling scheme in Algorithm 3. After marginalizing out $\dot{\boldsymbol{z}}$ and using the variable substitutions introduced in Equation (A.139), the joint density described by the graphical model can be written as

$$\begin{aligned}
p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{y}|\boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) &= p(\boldsymbol{o}|\boldsymbol{G})p(\boldsymbol{y}|\boldsymbol{\sigma}, \boldsymbol{o}, \boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{\phi}) \\
&= \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{y}|\boldsymbol{z} + \boldsymbol{o}, \boldsymbol{T})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}) \\
&= \mathcal{N}(\boldsymbol{o}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}})\mathcal{N}(\boldsymbol{o}|\boldsymbol{y} - \boldsymbol{z}, \boldsymbol{T})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}) \\
&= \mathcal{N}(\boldsymbol{o}|\boldsymbol{m}, \boldsymbol{C})\mathcal{N}(\boldsymbol{y} - \boldsymbol{z}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}} + \boldsymbol{T})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}) \\
&= \mathcal{N}(\boldsymbol{o}|\boldsymbol{m}, \boldsymbol{C})\mathcal{N}(\boldsymbol{z}|\boldsymbol{y}, \tilde{\boldsymbol{\Omega}} + \boldsymbol{T})\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{\phi}}) \\
&= \mathcal{N}(\boldsymbol{o}|\boldsymbol{m}, \boldsymbol{C})\mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \tilde{\boldsymbol{\Omega}} + \boldsymbol{T} + \boldsymbol{C}_{\boldsymbol{\phi}})\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z),
\end{aligned} \tag{A.151}$$

where

$$\boldsymbol{\mu}_z = \boldsymbol{\Sigma}_z(\tilde{\boldsymbol{\Omega}} + \boldsymbol{T})^{-1}\boldsymbol{y} \tag{A.152}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_z &= ((\tilde{\boldsymbol{\Omega}} + \boldsymbol{T})^{-1} + \boldsymbol{C}_{\boldsymbol{\phi}}^{-1})^{-1} \\
&= \boldsymbol{C}_{\boldsymbol{\phi}} - \boldsymbol{C}_{\boldsymbol{\phi}}(\tilde{\boldsymbol{\Omega}} + \boldsymbol{T} + \boldsymbol{C}_{\boldsymbol{\phi}})^{-1}\boldsymbol{C}_{\boldsymbol{\phi}} \\
&= (\tilde{\boldsymbol{\Omega}} + \boldsymbol{T})(\tilde{\boldsymbol{\Omega}} + \boldsymbol{T} + \boldsymbol{C}_{\boldsymbol{\phi}})^{-1}\boldsymbol{C}_{\boldsymbol{\phi}} \\
&= \boldsymbol{C}_{\boldsymbol{\phi}}(\tilde{\boldsymbol{\Omega}} + \boldsymbol{T} + \boldsymbol{C}_{\boldsymbol{\phi}})^{-1}(\tilde{\boldsymbol{\Omega}} + \boldsymbol{T}).
\end{aligned} \tag{A.153}$$

After marginalizing out $\boldsymbol{o}$ and dividing by the marginal of $\boldsymbol{y}$, we get the conditional distribution

$$p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\phi}, \boldsymbol{G}, \boldsymbol{\sigma}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z). \tag{A.154}$$

BIBLIOGRAPHY

[Lju98] Lennart Ljung. "System Identification". In: *Signal Analysis and Prediction*. Ed. by Ales Procházka, Jan Uhlíř, P. W. J. Rayner, and N. G. Kingsbury. Boston, MA: Birkhäuser Boston, 1998, 163.

[AM07] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[GPM89] Carlos E Garcia, David M Prett, and Manfred Morari. "Model predictive control: Theory and practice—A survey". In: *Automatica* 25.3 (1989), 335.

[Soh98] Björn Sohlberg. "Grey Box Modelling". In: *Supervision and Control for Industrial Processes: Using Grey Box Models, Predictive Control and Fault Detection Methods*. London: Springer London, 1998, 7.

[Zam+11] Elias Zamora-Sillero, Marc Hafner, Ariane Ibig, Joerg Stelling, and Andreas Wagner. "Efficient characterization of high-dimensional parameter spaces for systems biology". In: *BMC systems biology* 5.1 (2011), 1.

[Set09] Burr Settles. "Active learning literature survey". In: (2009).

[GF15] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), 1437.

[Bar74] Yonathan Bard. "Nonlinear parameter estimation". In: (1974).

[Ben79] M Benson. "Parameter fitting in dynamic models". In: *Ecological Modelling* 6.2 (1979), 97.

[Pon+62] L.S. Pontriagin, G. Boltjanskij, V.G. Boltânskij, Karreman Mathematics Research Collection, V. Gamkrelidze, K.N. Trirogoff, L.W. Neustadt, R.V. Gamkrelidze, W. Neustadt, E.F. Mišenko, et al. *The Mathematical Theory of Optimal Processes*. Interscience publishers. Interscience Publishers, 1962.

[Ma+21] Yingbo Ma, Vaibhav Dixit, Michael J Innes, Xingjian Guo, and Chris Rackauckas. "A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE. 2021, 1.

[Hou+12] Boris Houska, Filip Logist, Moritz Diehl, and Jan Van Impe. "A Tutorial on Numerical Methods for State and Parameter Estimation in Nonlinear Dynamic Systems". In: *Identification for Automotive Systems*. Ed. by Daniel Alberer, Håkan Hjalmarsson, and Luigi del Re. London: Springer London, 2012, 67.

[RC11] Christian Robert and George Casella. "A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data". In: *Statistical Science* (2011), 102.

[BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), 859.

[Var82] James M Varah. "A spline least squares method for numerical parameter estimation in differential equations". In: *SIAM Journal on Scientific and Statistical Computing* 3.1 (1982), 28.

[Wen+19] Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico S Gorbach, Andreas Krause, and Joachim M Buhmann. "Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, 1351.

[Wen+20] Philippe Wenk, Gabriele Abbati, Michael A Osborne, Bernhard Schölkopf, Andreas Krause, and Stefan Bauer. "Odin: Ode-informed regression for parameter and state inference in time-continuous dynamical systems". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, 6364.

[Ang+20] Emmanouil Angelis, Philippe Wenk, Bernhard Schölkopf, Stefan Bauer, and Andreas Krause. *SLEIPNIR: Deterministic and Provably Accurate Feature Expansion for Gaussian Process Regression with Derivatives*. 2020.

[Tre+21] Lenart Treven, Philippe Wenk, Florian Dörfler, and Andreas Krause. "Distributional Gradient Matching for Learning Uncertain Neural Dynamics Models". In: 2021.

[Abb+19] Gabriele Abbati, Philippe Wenk, Michael A. Osborne, Andreas Krause, Bernhard Schölkopf, and Stefan Bauer. "AReS and MaRS Adversarial and MMD-Minimizing Regression for SDEs". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, 1.

[BKS14] Ann C Babtie, Paul Kirk, and Michael PH Stumpf. "Topological sensitivity analysis for systems biology". In: *Proceedings of the National Academy of Sciences* 111.52 (2014), 18507.

[PBP18] Niklas Pfister, Stefan Bauer, and Jonas Peters. "Identifying Causal Structure in Large-Scale Kinetic Systems". In: *arXiv preprint arXiv:1810.11776* (2018).

[CGL09] Ben Calderhead, Mark Girolami, and Neil D Lawrence. "Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes". In: *Advances in neural information processing systems*. Citeseer. 2009, 217.

[Ram+07] Jim O Ramsay, Giles Hooker, David Campbell, and Jiguo Cao. "Parameter estimation for differential equations: a generalized smoothing approach". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.5 (2007), 741.

[Don+13] Frank Dondelinger, Dirk Husmeier, Simon Rogers, and Maurizio Filippone. "ODE parameter inference using adaptive gradient matching with Gaussian processes". In: *Artificial intelligence and statistics*. PMLR. 2013, 216.

[GBB17] Nico S Gorbach, Stefan Bauer, and Joachim M Buhmann. "Scalable Variational Inference for Dynamical Systems". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[WB14] Yali Wang and David Barber. "Gaussian Processes for Bayesian Estimation in Ordinary Differential Equations". In: *International Conference on Machine Learning (ICML)* (2014).

[MHH15] Benn Macdonald, Catherine Higham, and Dirk Husmeier. "Controversy in mechanistic modelling with Gaussian processes". In: *International Conference on Machine Learning*. 2015, 1539.

[Hin02] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), 1771.

[CS07] Taeryon Choi and Mark J Schervish. "On posterior consistency in nonparametric regression problems". In: *Journal of Multivariate Analysis* 98.10 (2007), 1969.

[Duv+13] David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. "Structure discovery in nonparametric regression through compositional kernel search". In: *International Conference on Machine Learning*. PMLR. 2013, 1166.

[Gor+17] Nico S Gorbach, Andrew An Bian, Benjamin Fischer, Stefan Bauer, and Joachim M Buhmann. "Model Selection for Gaussian Process Regression". In: *German Conference on Pattern Recognition*. Springer. 2017, 306.

[Duv14] David Duvenaud. "Automatic model construction with Gaussian processes". PhD thesis. University of Cambridge, 2014.

[Ras04a] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Advanced lectures on machine learning*. Springer, 2004, 63.

[Ras04b] Carl Edward Rasmussen. "Gaussian Processes in Machine Learning". In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 63.

[MD17] Benn Macdonald and Frank Dondelinger. *deGradInfer: Parameter Inference for Systems of Differential Equation*. `https://CRAN.R-project.org/package=deGradInfer`. 2017.

[Mac17] Benn Macdonald. "Statistical inference for ordinary differential equations using gradient matching". PhD thesis. University of Glasgow, 2017.

[Lot32] Alfred J Lotka. "The growth of mixed populations: two species competing for a common food supply". In: *Journal of the Washington Academy of Sciences* 22.16/17 (1932), 461.

[VG07] Vladislav Vyshemirsky and Mark A Girolami. "Bayesian ranking of biochemical system models". In: *Bioinformatics* 24.6 (2007), 833.

[Fit61] Richard FitzHugh. "Impulses and physiological states in theoretical models of nerve membrane". In: *Biophysical journal* 1.6 (1961), 445.

[NAY62] Jinichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. "An active pulse transmission line simulating nerve axon". In: *Proceedings of the IRE* 50.10 (1962), 2061.

[LF18] Marco Lorenzi and Maurizio Filippone. "Constraining the dynamics of deep probabilistic models". In: *International Conference on Machine Learning*. PMLR. 2018, 3227.

[GVW14] Javier González, Ivan Vujačić, and Ernst Wit. "Reproducing kernel Hilbert space based estimation of systems of ordinary differential equations". In: *Pattern Recognition Letters* 45 (2014), 26.

[Niu+16] Mu Niu, Simon Rogers, Maurizio Filippone, and Dirk Husmeier. "Fast inference in nonlinear dynamical systems using gradient matching". In: *Proceedings of Machine Learning Research* 48 (2016), 1699.

[SHS01] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. "A generalized representer theorem". In: *International conference on computational learning theory*. Springer. 2001, 416.

[Sol+03] Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J Leith, and Carl E Rasmussen. "Derivative observations in Gaussian process models of dynamic systems". In: *Advances in neural information processing systems*. 2003, 1057.

[LE98] Edward N Lorenz and Kerry A Emanuel. "Optimal sites for supplementary weather observations: Simulation with a small model". In: *Journal of the Atmospheric Sciences* 55.3 (1998), 399.

[QR05] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *Journal of Machine Learning Research* 6.Dec (2005), 1939.

[SG06] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian processes using pseudo-inputs". In: *Advances in neural information processing systems*. 2006, 1257.

[Tit09] Michalis Titsias. "Variational learning of inducing variables in sparse Gaussian processes". In: *Artificial Intelligence and Statistics*. 2009, 567.

[Wil+14] Andrew G Wilson, Elad Gilboa, Arye Nehorai, and John P Cunningham. "Fast kernel learning for multidimensional pattern extrapolation". In: *Advances in Neural Information Processing Systems*. 2014, 3626.

[CSS08] John P Cunningham, Krishna V Shenoy, and Maneesh Sahani. "Fast Gaussian process methods for point process intensity estimation". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, 192.

[WN15] Andrew Wilson and Hannes Nickisch. "Kernel interpolation for scalable structured Gaussian processes (KISS-GP)". In: *International Conference on Machine Learning*. 2015, 1775.

[SSH13] Simo Sarkka, Arno Solin, and Jouni Hartikainen. "Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering". In: *IEEE Signal Processing Magazine* 30.4 (2013), 51.

[WS01] Christopher KI Williams and Matthias Seeger. "Using the Nyström method to speed up kernel machines". In: *Advances in neural information processing systems*. 2001, 682.

[RR08] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems*. 2008, 1177.

[Laz+10] Miguel Lazaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Anibal R Figueiras-Vidal. "Sparse spectrum Gaussian process regression". In: *Journal of Machine Learning Research* 11.Jun (2010), 1865.

[HDS+17] James Hensman, Nicolas Durrande, Arno Solin, et al. "Variational Fourier Features for Gaussian Processes." In: *Journal of Machine Learning Research* 18.151 (2017), 1.

[MK18] Mojmir Mutny and Andreas Krause. "Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features". In: *Advances in Neural Information Processing Systems*. 2018, 9005.

[SS] Arno Solin and Simo Särkkä. "Hilbert space methods for reduced-rank Gaussian process regression". In: *Statistics and Computing* (), 1.

[Eri+18] David Eriksson, Kun Dong, Eric Lee, David Bindel, and Andrew G Wilson. "Scaling Gaussian process regression with derivatives". In: *Advances in Neural Information Processing Systems*. 2018, 6867.

[Sol+18] Arno Solin, Manon Kok, Niklas Wahlström, Thomas B Schön, and Simo Särkkä. "Modeling and interpolation of the ambient magnetic field by Gaussian processes". In: *IEEE Transactions on robotics* 34.4 (2018), 1112.

[SS19] Zoltán Szabó and Bharath Sriperumbudur. "On kernel derivative approximation with random Fourier features". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, 827.

[Rud91] Walter Rudin. "Functional analysis. 1991". In: *Internat. Ser. Pure Appl. Math* (1991).

[Hil87] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987.

[WF22] Jonas Wacker and Maurizio Filippone. "Local Random Feature Approximations of the Gaussian Kernel". In: *arXiv preprint arXiv:2204.05667* (2022).

[Boh06] Torsten P Bohlin. *Practical grey-box process identification: theory and applications*. Springer Science & Business Media, 2006.

[Kel+20] Jacob Kelly, Jesse Bettencourt, Matthew J Johnson, and David K Duvenaud. "Learning Differential Equations that are Easy to Solve". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, 4370.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[Dan+21] Raj Dandekar, Karen Chung, Vaibhav Dixit, Mohamed Tarek, Aslan Garcia-Valadez, Krishna Vishal Vemula, and Chris Rackauckas. "Bayesian neural ordinary differential equations". In: *48th ACM SIGPLAN Symposium on Principles of Programming Languages* (2021).

[Wil+16] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. "Deep kernel learning". In: *Artificial intelligence and statistics*. PMLR. 2016, 370.

[Cut+17] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. "Random feature expansions for deep Gaussian processes". In: *International Conference on Machine Learning*. PMLR. 2017, 884.

[Sch+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, 1889.

[Hou+16] Rein Houthooft, Xi Chen, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. "VIME: Variational Information Maximizing Exploration". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[Ber+17] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. "Safe Model-based Reinforcement Learning with Stability Guarantees". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[Kan39] Leonid V Kantorovich. "The mathematical method of production planning and organization". In: *Management Science* 6.4 (1939), 363.

[PD10] Gianluigi Pillonetto and Giuseppe De Nicolao. "A new kernel-based approach for linear system identification". In: *Automatica* 46.1 (2010), 81.

[WT11] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer. 2011, 681.

[CFG14] Tianqi Chen, Emily Fox, and Carlos Guestrin. "Stochastic gradient hamiltonian monte carlo". In: *International conference on machine learning*. PMLR. 2014, 1683.

[HG14] Matthew D Hoffman and Andrew Gelman. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *J. Mach. Learn. Res.* 15.1 (2014), 1593.

[Nor+21] Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. "Neural {ODE} Processes". In: *International Conference on Learning Representations*. 2021.

[Bra+18] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018.

[RR+07] Ali Rahimi, Benjamin Recht, et al. "Random Features for Large-Scale Kernel Machines." In: *NIPS*. Vol. 3. 4. Citeseer. 2007, 5.

[Liu+20] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. "When Gaussian process meets big data: A review of scalable GPs". In: *IEEE transactions on neural networks and learning systems* 31.11 (2020), 4405.

[MWV15] MJ Menne, CN Williams Jr, and RS Vose. *Long-term daily climate records from stations across the contiguous United States*. 2015.

[De +19] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series". In: *Advances in neural information processing systems* 32 (2019).

[Pic07] Umberto Picchini. "SDE Toolbox: Simulation and estimation of stochastic differential equations with MATLAB." In: (2007).

[BMR15] Harish S Bhat, RWMA Madushani, and Shagun Rawat. "Parameter inference for stochastic differential equations with density tracking by quadrature". In: *International Workshop on Simulation*. Springer. 2015, 99.

[Ryd+18] Thomas Ryder, Andrew Golightly, A Stephen McGough, and Dennis Prangle. "Black-box Variational Inference for Stochastic Differential Equations". In: *International Conference on Machine Learning* (2018).

[TS19] Filip Tronarp and Simo Särkkä. "Iterative statistical linear regression for Gaussian smoothing in continuous-time non-linear stochastic dynamic systems". In: *Signal Processing* 159 (2019), 1.

[Søro4] Helle Sørensen. "Parametric inference for diffusion processes observed at discrete points in time: a survey". In: *International Statistical Review* 72.3 (2004), 337.

[NMY00] Jan Nygaard Nielsen, Henrik Madsen, and Peter C Young. "Parameter estimation in stochastic differential equations: an overview". In: *Annual Reviews in Control* 24 (2000), 83.

[HJL07] A Stan Hurn, JI Jeisman, and Kenneth A Lindsay. "Seeing the wood for the trees: A critical evaluation of methods to estimate the parameters of stochastic differential equations". In: *Journal of Financial Econometrics* 5.3 (2007), 390.

[ECS01] Ola Elerian, Siddhartha Chib, and Neil Shephard. "Likelihood inference for discretely observed nonlinear diffusions". In: *Econometrica* 69.4 (2001), 959.

[Era01] Bjørn Eraker. "MCMC analysis of diffusion models with application to finance". In: *Journal of Business & Economic Statistics* 19.2 (2001), 177.

[PF20] Susanne Pieschner and Christiane Fuchs. "Bayesian inference for diffusion processes: using higher-order approximations for transition densities". In: *Royal Society open science* 7.10 (2020), 200270.

[MS+17] Frank van der Meulen, Moritz Schauer, et al. "Bayesian estimation of discretely observed multi-dimensional diffusion processes using guided proposals". In: *Electronic Journal of Statistics* 11.1 (2017), 2358.

[Sär+15] Simo Särkkä, Jouni Hartikainen, Isambi Sailon Mbalawata, and Heikki Haario. "Posterior inference on parameters of stochastic differential equations via non-linear Gaussian filtering and adaptive MCMC". In: *Statistics and Computing* 25.2 (2015), 427.

[Arc+07] Cedric Archambeau, Dan Cornford, Manfred Opper, and John Shawe-Taylor. "Gaussian process approximations of stochastic differential equations". In: *Journal of machine learning research* 1 (2007), 1.

[VOC15] Michail D Vrettas, Manfred Opper, and Dan Cornford. "Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations". In: *Physical Review E* 91.1 (2015), 012148.

[HL99] AS Hurn and KA Lindsay. "Estimating the parameters of stochastic differential equations". In: *Mathematics and computers in simulation* 48.4-6 (1999), 373.

[Ait02] Yacine Ait-Sahalia. "Maximum likelihood estimation of discretely sampled diffusions: a closed-form approximation approach". In: *Econometrica* 70.1 (2002), 223.

[RBO13] Andreas Ruttor, Philipp Batz, and Manfred Opper. "Approximate Gaussian process inference for the drift function in stochastic differential equations". In: *Advances in Neural Information Processing Systems*. 2013, 2040.

[Yil+18] Cagatay Yildiz, Markus Heinonen, Jukka Intosalmi, Henrik Mannerstrom, and Harri Lahdesmaki. "Learning stochastic differential equations with gaussian processes without gradient matching". In: *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2018, 1.

[RNR16] Christoph Riesinger, Tobias Neckel, and Florian Rupp. "Solving random ordinary differential equations on gpu clusters using multiple levels of parallelism". In: *SIAM Journal on Scientific Computing* 38.4 (2016), C372.

[Dos77] Halim Doss. "Liens entre équations différentielles stochastiques et ordinaires". In: (1977).

[Sus78] Héctor J Sussmann. "On the gap between deterministic and stochastic ordinary differential equations". In: *The Annals of Probability* (1978), 19.

[Bau+17] Stefan Bauer, Nico S Gorbach, Djordje Miladinovic, and Joachim M Buhmann. "Efficient and Flexible Inference for Stochastic Systems". In: *Advances in Neural Information Processing Systems*. 2017, 6988.

[KM18] Hadiseh Karimi and Kimberley B McAuley. "Bayesian Objective Functions for Estimating Parameters in Nonlinear Stochastic Differential Equation Models with Limited Data". In: *Industrial & Engineering Chemistry Research* 57.27 (2018), 8946.

[Jim+08] MJ Jiménez, Henrik Madsen, JJ Bloem, and Bernd Dammann. "Estimation of non-linear continuous time models for the heat exchange dynamics of building integrated photovoltaic modules". In: *Energy and Buildings* 40.2 (2008), 157.

[DS13] Sophie Donnet and Adeline Samson. "A review on estimation of stochastic differential equations for pharmacokinetic/pharmacodynamic models". In: *Advanced Drug Delivery Reviews* 65.7 (2013), 929.

[Ras04c] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Advanced lectures on machine learning*. Springer, 2004, 63.

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, 2672.

[YZK20] Liu Yang, Dongkun Zhang, and George Em Karniadakis. "Physics-informed generative adversarial networks for stochastic differential equations". In: *SIAM Journal on Scientific Computing* 42.1 (2020), A292.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International Conference on Machine Learning*. 2017, 214.

[DRG15] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. "Training generative neural networks via maximum mean discrepancy optimization". In: *arXiv preprint arXiv:1505.03906* (2015).

[Gre+12] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. "A kernel two-sample test". In: *Journal of Machine Learning Research* 13.Mar (2012), 723.

[LSZ15] Yujia Li, Kevin Swersky, and Rich Zemel. "Generative moment matching networks". In: *International Conference on Machine Learning*. 2015, 1718.

[PP+08] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. "The matrix cookbook". In: *Technical University of Denmark* 7.15 (2008), 510.

[Lor63] Edward N Lorenz. "Deterministic nonperiodic flow". In: *Journal of the atmospheric sciences* 20.2 (1963), 130.

[Bez+17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1 (2017), 65.

PUBLICATIONS

Conference contributions:

[Wen+19] Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico S Gorbach, Andreas Krause, and Joachim M Buhmann. "Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, 1351.

[Wen+20] Philippe Wenk, Gabriele Abbati, Michael A Osborne, Bernhard Schölkopf, Andreas Krause, and Stefan Bauer. "Odin: Ode-informed regression for parameter and state inference in time-continuous dynamical systems". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, 6364.

[Abb+19] Gabriele Abbati, Philippe Wenk, Michael A. Osborne, Andreas Krause, Bernhard Schölkopf, and Stefan Bauer. "AReS and MaRS Adversarial and MMD-Minimizing Regression for SDEs". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, 1.

[Tre+21] Lenart Treven, Philippe Wenk, Florian Dörfler, and Andreas Krause. "Distributional Gradient Matching for Learning Uncertain Neural Dynamics Models". In: 2021.

[Cal+22] Edoardo Caldarelli, Philippe Wenk, Stefan Bauer, and Andreas Krause. "Adaptive Gaussian Process Change Point Detection". In: *International Conference on Machine Learning*. PMLR. 2022, 2542.

[Sch+21] Andreas Schlaginhaufen, Philippe Wenk, Andreas Krause, and Florian Dorfler. "Learning Stable Deep Dynamics Models for Partially Observed or Delayed Dynamical Systems". In: vol. 34. 2021, 11870.

[Agu+20] Diego Agudelo-Espana, Andrii Zadaianchuk, Philippe Wenk, Aditya Garg, Joel Akpo, Felix Grimminger, Julian Viereck, Maximilien Naveau, Ludovic Righetti, Georg Martius, et al. "A real-robot dataset for assessing transferability of learned dynamics models". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, 8151.

Preprints:

[Ang+20] Emmanouil Angelis, Philippe Wenk, Bernhard Schölkopf, Stefan Bauer, and Andreas Krause. *SLEIPNIR: Deterministic and Provably Accurate Feature Expansion for Gaussian Process Regression with Derivatives*. 2020.