Bayesian Learning for Control in Multimodal Dynamical Systems

By Aidan J. Scannell

Department of Aerospace Engineering
University of Bristol
&
Department of Engineering Design and Mathematics
University of the West of England

A dissertation submitted to the University of Bristol and the University of the West of England in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering.

Supervisors: Professor Arthur Richards
Dr Carl Henrik Ek

May 2022

Abstract

Over the last decade, *learning-based control* has become a popular paradigm for controlling dynamical systems. Although recent algorithms can find high-performance controllers, they typically only consider unimodal systems and cannot correctly identify multimodal dynamical systems. The main goal of this thesis is to control *unknown*, *multimodal* dynamical systems, to a target state, whilst *avoiding inoperable or undesirable dynamics modes*. Further to this, deploying learning algorithms in the real world requires handling the uncertainties inherent to the system, as well as the uncertainties arising from learning from observations. To this end, we consider the Model-Based Reinforcement Learning (MBRL) setting, where an explicit dynamics model – that includes uncertainties – is used to plan trajectories to a target state.

Motivated by synergising model learning and control, we introduce a Mixtures of Gaussian Process Experts (MoGPE) method for learning dynamics models, which infers latent structure regarding how systems switch between their underlying dynamics modes. We then present three trajectory optimisation algorithms which, given this learned dynamics model, find trajectories to a target state with *mode remaining guarantees*. Initially, the agent's dynamics model will be highly *uncertain* — due to a lack of training observations — so these algorithms cannot guarantee mode remaining navigation with high confidence. When this is the case, the agent actively explores its environment, collects data and updates its dynamics model. We introduce an explorative trajectory optimisation algorithm that explicitly reasons about the uncertainties in the dynamics model. As a result, it can explore the environment whilst guaranteeing that the agent remains in the desired dynamics mode with high probability. Finally, we consolidate the work in this thesis into a MBRL algorithm, which solves the mode remaining navigation problem, whilst guaranteeing that the controlled system remains in the desired dynamics mode with a high probability.

Covid-19 Statement

To mitigate risk due to Covid-19 lab closures, many of the methods in this thesis are validated in simulated experiments and not in real-world experiments. A real-world state transition data set was collected by flying a quadcopter around the Bristol Robotics Laboratory (BRL) with constant controls. This data set was used to test the MoSVGPE method in Section 3.5.3 and to test the collocation solver presented in Section 4.2.2. These results are a step towards validating that the methods presented in this thesis work on real-world systems. However, this constant controls data set could not be used to learn a dynamics model for the control methods in Sections 4.2.3 and 4.3 (due to the lack of controls). Therefore, to mitigate the risk associated with Covid-19 lab closures, we decided not to collect a new real-world data set that could be used to train the dynamics model. Instead, we developed a simulator and used it to generate a data set. This had the added benefit that the simulator could be used to test the trajectory optimisation algorithms presented in Sections 4.2.3 and 4.3 and Chapter 6.

Declaration

I declare that the work in this thesis was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the thesis are those of the author.

SIGNED: ..... DATE: 12th May 2022

Acknowledgements

I am deeply grateful to my two supervisors, Arthur Richards and Carl Henrik Ek. I have learned so much under your supervision and am incredibly grateful for your continued support. Your expertise and willingness to explore new ideas have made my PhD journey enjoyable. Arthur, I have especially enjoyed the many connections you've made between concepts from machine learning and control theory. You have shown me how to think deeply about why things work well and how to build rich intuitions for seemingly complex problems. Carl, thank you for the countless hours you spent teaching me machine learning theory. You are a talented and inspiring teacher, and I wholeheartedly appreciate your advice and enthusiasm over the years. I have thoroughly enjoyed our meetings, especially those that derailed into discussions about bikes and sharing Emacs lisp snippets. I could not have asked for better supervisors. You have both inspired me.

I want to thank my friends and colleagues in Bristol for their support and for making my time in Bristol so much fun! I also want to thank my family: mum, dad, Ciaran and Mhairi, for being amazing; none of this would have been possible without your support. I love you all very much.

I have thoroughly enjoyed writing this thesis in Org Mode. I would like to thank everyone that helped me get started with Emacs, and I would especially like to thank all of the contributors to GNU Emacs and Org Mode.

Finally, I would like to thank the Reddit users that supported me through my journey with RSI. It seems like a lifetime ago since I was considering giving up my PhD along with any hopes of a career involving programming. Your support gave me the confidence that I needed to overcome my RSI. I am eternally grateful.

This work was supported by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE) at the Bristol Robotics Laboratory (BRL).

Contents

List of Tables

List of Figures

Acronyms

ARD Automatic Relevance Determination. 23, 53, 112, 151

BALD Bayesian Active Learning by Disagreement. 165, 166

BRL Bristol Robotics Laboratory. iii, vii, 64

CaI Control as Inference. 76, 97, 101, 172

DRE Direct Optimal Control via Riemannian Energy. x, 78, 90, 107, 109, 110, 120, 121, 124–128, 130–138, 140, 172

ELBO Evidence Lower Bound. 40, 41, 44, 53, 63, 73, 89, 90, 104–106, 122, 136, 137, 140, 149

FARSCOPE Future Autonomous and Robotic Systems. vii

FITC Fully Independent Training Conditional. 43

GP Gaussian Process. 5, 15, 17, 20, 22, 24–26, 28, 29, 31, 32, 35–45, 47, 51, 53, 54, 57, 58, 60, 62, 67, 68, 70–72, 77, 80, 82–85, 90, 93, 100, 103, 106, 111, 112, 116–118, 120–126, 129, 130, 133, 136, 139, 145, 147, 148, 151, 153, 154, 156, 157, 160, 161, 164, 167, 168, 172, 173

GPs Gaussian Processes. 15, 20–22, 24, 26, 28, 33, 34, 39, 42, 46, 53, 56, 59, 68, 72, 73, 133, 138, 139, 149, 164, 167

GPSSM Gaussian Process State Space Model. 30

IG Indirect Optimal Control via Latent Geodesics. x, xiii, xvi, 76, 78, 86, 107, 109–111, 113, 120–127, 130, 131, 134, 136–138, 140

iLQG iterative Linear Quadratic Gaussian. 100, 139

iLQR iterative Linear Quadratic Regulator. 15, 100

LQR Linear Quadratic Regulator. 15

MAE Mean Absolute Error. 52–54

MBRL Model-Based Reinforcement Learning. i, 12–14, 20, 26–28, 72, 139, 142, 149, 163, 172, 173

MDP Markov Decision Process. xv, 8, 10, 11

MFRL Model-Free Reinforcement Learning. 13, 174

ModeOpt Mode Optimisation. xvi, xvii, 142, 145, 146, 149–152, 154–169, 171–173

MoE Mixture of Experts. 34, 35, 51

MoGPE Mixtures of Gaussian Process Experts. i, 5, 26, 31, 33–35, 37, 51, 52, 68, 70–72, 97, 107, 139, 164, 166

MoSVGPE Mixtures of Sparse Variational Gaussian Process Experts. iii, xiii, xv, xvi, 31, 53–57, 59, 63, 75, 77, 79, 80, 92, 94, 97, 101, 107, 110, 111, 117, 118, 139, 140, 145, 149–151, 159, 166, 167, 171

MPC Model Predictive Control. 13, 15, 17, 18, 20, 28, 139, 164, 174

MRCaI Mode Remaining Control as Inference. 97, 107, 109, 110, 120, 121, 124–126, 128–130, 132, 133, 135–140, 172

NLPP Negative Log Predictive Probability. 52–54, 60, 71

ODE Ordinary Differential Equation. 78, 86–88, 90, 114

PETS Probabilistic Ensembles with Trajectory Sampling. 28

PID Proportional Integral Derivative. 64, 65, 69

PILCO Probabilistic Inference for Learning COntrol. 28

PIPPS Probabilistic Inference for Particle-based Policy Search. 28

RL Reinforcement Learning. 1, 7, 8, 11–13, 18–20, 27, 30

RMSE Root Mean Squared Error. 52–54

SDE Stochastic Differential Equation. 90

SLSQP Sequential Least Squares Programming. 89, 96

SOC Stochastic Optimal Control. 7, 90, 99, 100

SVGP Sparse Variational Gaussian Process. 53–57, 59, 63, 89

SVI Stochastic Variational Inference. 40, 43

VAE Variational Auto-Encoder. 26

1 Introduction

The modern world is pervaded with dynamical systems that we seek to control to achieve a desired behaviour. Examples include autonomous vehicles, aircraft, robotic manipulators, financial markets and energy management systems. In the last decade, Reinforcement Learning (RL), and learning-based control in general, have become popular paradigms for controlling dynamical systems (Hewing et al., 2020b; Sutton and Barto, 2018). This can be accounted to significant improvements in sensing and computational capabilities, as well as recent successes in machine learning.

This growing interest in learning-based control has emphasised the importance of real-world considerations. Real-world systems are often highly *nonlinear*, exhibit *stochasticity* and *multimodalities*, are expensive to run (energy-intensive, subject to wear and tear) and must be controlled subject to *constraints* (for safety, efficiency, and so on). In contrast to simulation, the control of physical systems also has real-world consequences: components may get damaged, the system may damage its environment, or the system may catastrophically fail. As such, any learning-based control strategy deployed in the real world should handle both the uncertainty inherent to the environment and the uncertainty introduced by learning from observations.

Many dynamical systems exhibit *multimodalities*, where some of the dynamics modes are believed to be *inoperable* or *undesirable*. These *multimodalities* may be due to spatially varying model parameters, for example, process noise terms modelling aircraft turbulence, or friction coefficients modelling surface-tyre interactions for

ground vehicles over different terrain. In these systems, it is desirable to avoid entering specific dynamics modes that are believed to be *inoperable*. Perhaps they are hard to control due to stability, or the mode switch is hard to control. Alternatively, they may be *inefficient* or low *performing*. Given these motivations, this thesis focuses on controlling dynamical systems from an initial state, to a target state, whilst avoiding specific dynamics modes.

Model-based control comprises a powerful set of techniques for finding controls of *constrained* dynamical systems, given a dynamics model describing the evolution of the controlled system. It is commonly used for controlling aircraft, robotic manipulators, and walking robots (Betts, 1998; Garg et al., 2010; Von Stryk and Bulirsch, 1992). One caveat is that it requires a relatively accurate mathematical model of the system. Traditionally, these mathematical models are built using first principles based on physics. However, accurately modelling the underlying transition dynamics can be challenging and lead to the introduction of model errors. These model errors may be due to incorrectly specifying model parameters or the models themselves. For example, modelling a nonlinear system to be linear or a multimodal system to be unimodal. Incorrectly specifying these model parameters and their associated uncertainty can have a detrimental impact on controller performance. Handling these issues is a central goal of robust and stochastic optimal control (Freeman and Kokotovic, 1996; Stengel, 1986).

The difficulties associated with constructing mathematical representations of dynamical systems can be overcome by learning from observations (Ljung, 1999). Learning dynamics models has the added benefit that it alleviates the dependence on domain experts for specifying accurate models, making it easier to deploy more general techniques. However, learning dynamics models for control introduces other difficulties. For example, it is important to know where the model cannot predict confidently due to a lack of training observations. This concept is known as *epistemic uncertainty* and is reduced in the limit of infinite data. Correctly quantifying uncertainty is crucial for intelligent decision-making.

Epistemic uncertainty is the uncertainty attributed to incomplete knowledge about a phenomenon that limits our ability to model it. It represents knowledge about a phenomenon that we could know but we do *not know a priori*. It is reduced through the accumulation of additional information.

In a **risk-averse setting**, control strategies should avoid entering regions of a learned dynamics model with high *epistemic uncertainty*. This is because it is impossible to guarantee *constraint* satisfaction in a learned model, i.e. if the trajectory will avoid the undesired dynamics mode. Conversely, in an **explorative setting**, if the *epistemic uncertainty* has been quantified, it can be used to guide exploration into regions of the dynamics that have not previously been observed. This experience can then be used to update the model, in turn reducing its *epistemic uncertainty*.

These two settings are the main focus of this thesis. If the dynamics are not fully *known a priori*, an agent will not be able to plan a risk-averse trajectory to the target state confidently. How can the agent explore its environment, in turn reducing the *epistemic uncertainty* associated with its dynamics model? As this thesis assumes that complete knowledge of the dynamics are *not fully known a priori*, a main interest is jointly inferring the underlying dynamics modes, as well as how the system switches between them, through repeated interactions with the system. Once the agent has explored enough, how can this learned dynamics model be exploited to plan risk-averse trajectories that remain in the desired dynamics mode?

1.1 Illustrative Example

The methods developed throughout this thesis are motivated by a 2D quadcopter navigation example. See Figure 1.1 for a schematic of the environment and details of the problem. The goal is to fly the quadcopter from an initial state $\mathbf{x}_0$, to a target state $\mathbf{x}_f$. However, it considers a quadcopter operating in an environment subject to spatially varying wind – induced by a fan – where two dynamics modes can represent the system,

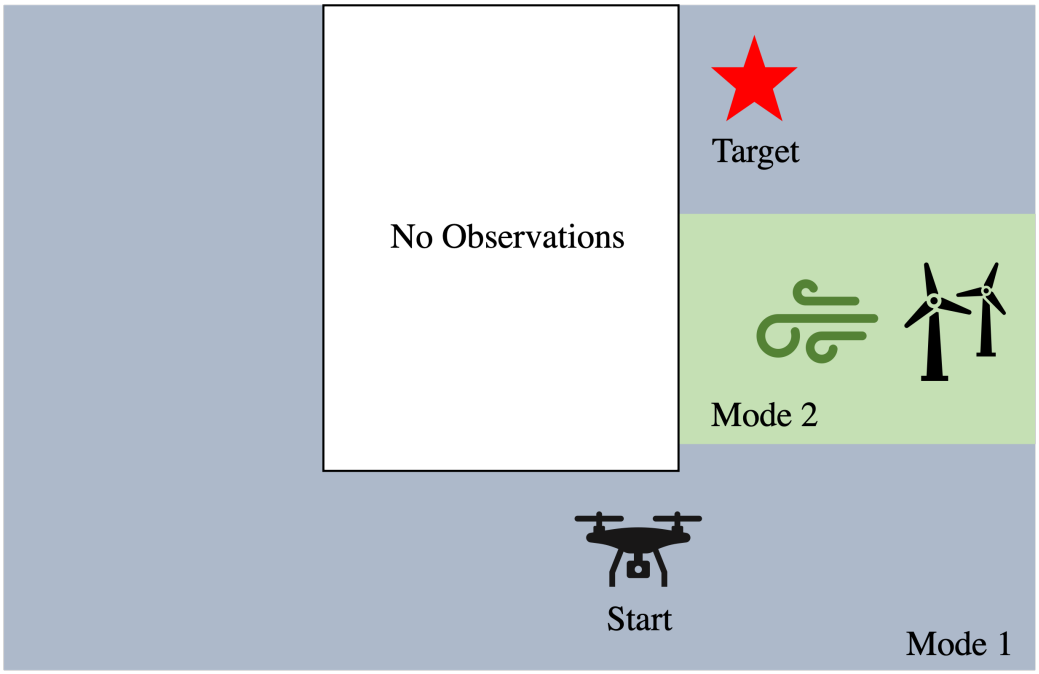


Figure 1.1: ***Quadcopter navigation problem*** *Diagram showing a top-down view of an environment, representing a quadcopter subject to two dynamics modes: 1) an operable dynamics mode away from the fan (blue) and 2) an inoperable, turbulent dynamics mode in front of the fan (green). The goal is to find trajectories from a start state* $\mathbf{x}_0$*, to the target state* $\mathbf{x}_f$ *(red star), whilst avoiding the turbulent dynamics mode.*

Mode 1 is an *operable* dynamics mode away from the fan,

Mode 2 is an *inoperable*, turbulent dynamics mode in front of the fan.

The turbulent dynamics mode is subject to higher drift (in the negative x direction) and to higher diffusion (process noise). It is hard to know the exact turbulent dynamics due to complex and uncertain interactions between the quadcopter and the wind field. Further to this, controlling the system in the turbulent dynamics mode may be infeasible. This is because the unpredictability of the turbulence may cause catastrophic failure. Therefore, when flying the quadcopter from an initial state $\mathbf{x}_0$, to a target state $\mathbf{x}_f$, it is desirable to find trajectories that avoid entering this turbulent dynamics mode. The state-space of the velocity controlled quadcopter example consists of the 2D Cartesian coordinates $\mathbf{x} = (x, y)$ and the controls consist of the speed in each direction, given by $\mathbf{u} = (v_x, v_y)$.

1.2 Contributions

This thesis explores methods for mode remaining control in multimodal dynamical systems that explicitly reason about the uncertainties arising during learning and control. The primary contributions of this thesis are as follows:

- Chapter 3: is concerned with learning representations of multimodal dynamical systems, where both the underlying dynamics modes and how the system switches between them, are *not fully known a priori*. Motivated by learning dynamics models for model-based control, it formulates a probabilistic model rich with latent spaces for control. It then derives a variational inference scheme that principally handles uncertainty whilst providing scalability via stochastic gradient methods. The method is a Mixtures of Gaussian Process Experts (MoGPE) method with a Gaussian Process (GP)-based gating network.
- Chapter 4: investigates model-based control techniques that leverage the probabilistic model from Chapter 3 to solve the mode remaining navigation problem. Due to the complexity of the problem, this chapter assumes prior access to the environment, such that a data set of state transitions $\mathcal{D}$ has previously been collected and used to train the model. It presents three trajectory optimisation algorithms that leverage the learned dynamics model's latent structure to solve the mode remaining navigation problem.
- Chapter 6: then considers the more realistic scenario of not having prior access to the environment. In this scenario, the agent does not have access to a historical data set for model learning. Instead, it must actively explore its environment, collect data and use it to update its dynamics model, whilst simultaneously attempting to remain in the desired dynamics mode. It presents an exploration strategy for exploring multimodal dynamical systems whilst remaining in a desired dynamics mode with high probability. It then details how this exploration strategy can be combined with the methods from Chapters 3 and 4 to solve the mode remaining navigation problem.

1.3 Associated Publications

The first trajectory optimisation algorithm presented in Section 4.2.2 and an initial version of the approach for learning multimodal dynamical systems in Chapter 3, are published in:

Aidan Scannell et al. (2021). "Trajectory Optimisation in Learned Multimodal Dynamical Systems Via Latent-ODE Collocation". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE

2 Background and Related Work

The primary goal of this thesis is to control *stochastic*, *multimodal*, *nonlinear* dynamical systems to a target state, whilst remaining in the desired dynamics mode. This is a Stochastic Optimal Control (SOC) problem which can be summarised as follows:

> *For a given multimodal dynamical system with control inputs, determine a controller that can navigate to a target state, whilst remaining in a desired dynamics mode.*

This chapter formally defines this mode remaining navigation problem and reviews the relevant literature.

2.1 Problem Statement

Dynamical systems describe the behaviour of a system over time t and are a key component of both control theory and RL. At any given time t, a dynamical system has a state, represented as a vector of real numbers $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^{D_x}$. The system can be controlled by applying control actions $\mathbf{u}_t \in \mathcal{U} \subseteq \mathbb{R}^{D_u}$ at any given time. This thesis considers *stochastic*, *multimodal*, *nonlinear* dynamical systems, given by,

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon} && \text{(2.1a)} \\ &= f_k(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon}_k \quad \text{if} \quad \alpha(\mathbf{x}_t) = k \\ \boldsymbol{\epsilon}_k &\sim \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\epsilon}_k}), && \text{(2.1b)} \end{aligned}$$

where the discrete mode indicator function $\alpha : \mathcal{X} \to \mathcal{A}$ indicates which of the K underlying dynamics modes $\{f_k : \mathcal{X}_k \times \mathcal{U} \to \mathcal{X}\}_{k=1}^{K}$, and associated noise models $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_k})$, governs the system at a given time step t. The output of the mode indicator function is referred to as the mode indicator variable and is given by $\alpha_t = \alpha(\mathbf{x}_t) \in \mathcal{A} = \{1, \ldots, K\} = \mathbb{Z} \cap [1, K]$.

This thesis assumes that the state $\mathbf{x}$ is observed directly and is not subject to observation noise. This is a standard assumption in the Markov Decision Process (MDP) framework, which is commonly adopted in the RL literature. In this case, the $\boldsymbol{\epsilon}_k$ term solely represents the process noise, which accounts for unwanted and, in general, unknown system disturbances. For example, it is hard to model aerodynamic effects on aircraft, so these could be accounted for in the process noise term.

Optimal control Optimal control is a branch of mathematical optimisation that seeks to find a controller π that optimises an objective function $J_\pi(\mathbf{x})$. The objective function might be formulated to solve a particular task or to make the dynamical system behave in a certain way. Typically the goal is to minimise a cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. This thesis considers the *finite horizon problem*, given by,

$$J_\pi(\mathbf{x}) = \mathbb{E}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \mid \mathbf{x}_0 = \mathbf{x}\right] \tag{2.2}$$

where T is known as the horizon. The objective $J_\pi(\mathbf{x})$ quantifies the cost of deploying the controller π from an initial state $\mathbf{x}$. The cost function typically consists of a terminal cost and a term which is integrated over the horizon (integral cost). In a navigation task, the terminal cost may consist of the distance to the target, whilst the integral cost may encode the notion of minimum effort control, e.g. energy consumption (Kirk, 2004).

Controller space The controller space Π defines the set of controllers over which optimisation is performed. Controllers can have state feedback such that they are given by $\mathbf{u}_t = \pi(\mathbf{x}_t)$. These are closed-loop controllers and in the RL literature are referred to as policies. Alternatively, controllers can depend on time $\mathbf{u}_t = \pi(t)$, in

which case they are open-loop controllers. This thesis considers the general case, given by $\mathbf{u}_t = \pi(\mathbf{x}_t, t)$, which encompasses both open-loop and closed-loop controllers.

Mode remaining This thesis considers systems where the underlying dynamics modes are defined by disjoint state domains, i.e. $\mathcal{X}_k = \{\mathbf{x} \in \mathcal{X} \mid \alpha(\mathbf{x}) = k\}$, with $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for distinct $i, j \in \{1, \dots K\}$. Notice that each mode's dynamics can leave their state space $\mathcal{X}_k$ and enter another mode. Ideally, this work seeks to enforce the controlled system to remain in a given mode. Formally, a mode remaining controlled system is defined as follows.

Definition 2.1.1 (Mode Remaining)**.** *Let k denote a dynamics mode, defined by its state domain $\mathcal{X}_k \subseteq \mathcal{X}$. Given an initial state $\mathbf{x}_0 \in \mathcal{X}_k$, and a controller $\pi \in \Pi$, the controlled system is said to be mode-remaining iff:*

$$f(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \in \mathcal{X}_k \quad \forall t \tag{2.3}$$

Mode remaining navigation problem Given this definition of a mode remaining controlled system, this thesis seeks to solve,

$$\min_{\pi \in \Pi} \quad \mathbb{E}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \mid \mathbf{x}_0 = \mathbf{x}_0\right] \tag{2.4a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f_k(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) + \boldsymbol{\epsilon}_k, \quad \alpha(\mathbf{x}_t) = k \qquad \forall t \in \{0, \dots, T-1\} \tag{2.4b}$$

$$f(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \in \mathcal{X}_{k^*} \qquad \forall t \in \{0, \dots, T-1\} \tag{2.4c}$$

$$\mathbf{x}_0 = \mathbf{x}_0 \tag{2.4d}$$

$$\mathbf{x}_T = \mathbf{x}_f, \tag{2.4e}$$

where $\mathbf{x}_f$ denotes the target state and k^* denotes the desired dynamics mode.

Note that the objective function in Equation (2.2) is not of primary interest in this work. The novelty of this work arises from remaining in the desired dynamics mode k^*.

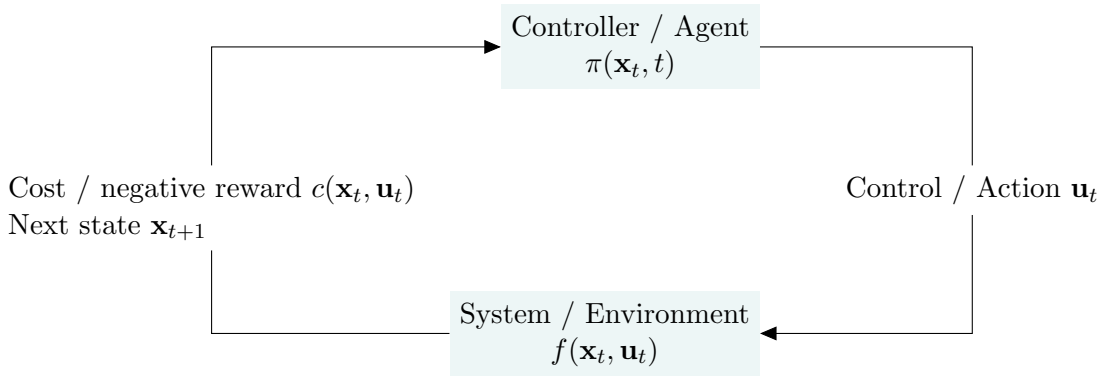


Figure 2.1: ***Markov Decision Process (MDP)*** *Illustration of an MDP through the lens of optimal control. The controller (agent) uses all past knowledge, absorbed into the previous state $\mathbf{x}_t$ (Markov property), to decide which control (action) $\mathbf{u}_t$ to execute. It then observes the next state $\mathbf{x}_{t+1}$ and the cost (negative reward) associated with the state transition. The cost function is assumed to be known. The agent's goal is to minimise the cumulative cost (maximise the cumulative reward).*

2.2 Optimal Control

The discrete-time optimal control problem considered in this thesis can be modelled as a Markov Decision Process (MDP), as seen in Figure 2.1. The MDP framework refers to the controller π as the agent, the control $\mathbf{u}$ as the action, and the dynamical system f as the environment. Further to this, the goal is to maximise the cumulative reward instead of minimising the cumulative cost. However, slight manipulation of the objective enables cost and reward functions to be interchanged. The key idea in MDPs is that all past information is represented by the current state $\mathbf{x}_t$, which is known as the Markov property. In this thesis, most work is formulated through the lens of optimal control. However, the controller is sometimes referred to as the agent and the system is often referred to as the environment.

In general, solutions to optimal control problems can be characterised by two different approaches, Pontryagin's Minimum Principe (Pontryagin, 1987) (based on the calculus of variations) and Bellman's Minimum Principle (Bellman, 1956). Although Pontryagin's Minimum Principle offers computational benefits over Bellman's principle, it does not readily generalise to the stochastic case. For this reason, we restrict

our discussion to Bellman's principle and the solution arising from it, known as dynamic programming.

2.2.1 Dynamic Programming

Dynamic programming (Bellman, 1956) encompasses a large class of algorithms that can be used to find optimal controllers given a model of the environment as an MDP. However, classical dynamic programming algorithms are of limited use as they rely on accurate dynamics models and have a significant computational expense. Nevertheless, they are still important theoretically. The main idea of dynamic programming (and RL in general) is to structure the search for good controllers using value functions V. Optimal controllers can easily be found from optimal value functions V_*, which satisfy the Bellman optimality equations,

$$V_*(\mathbf{x}) = \min_{\mathbf{u}} \mathbb{E}\left[\underbrace{c(\mathbf{x}_t, \pi(\mathbf{x}_t, t))}_{\text{first step}} + \underbrace{V_*(f(\mathbf{x}_t, \pi(\mathbf{x}_t, t)))}_{\text{future}} \mid \mathbf{x}_0 = \mathbf{x}\right]. \tag{2.5}$$

In the finite horizon setting, directly solving the Bellman equations backwards in time is referred to as dynamic programming.

Approximate dynamic programming encompasses a large class of methods that, given a controller π, approximate the value function $V_\pi(\mathbf{x})$ for each state $\mathbf{x}$ with a parameterised function. The approximate value function can then be used for *policy improvement*, where a controller with superior performance is computed. In general, approximate dynamic programming is a large collection of algorithms that encompasses methods from RL as well. However, these methods are out of the scope of this thesis.

2.2.2 Reinforcement Learning

There are multiple approaches to finding controllers π that minimise the expected cost in Equation (2.2) subject to the stochastic dynamics in Equation (2.1). How-

Algorithm 1 Model-Based Reinforcement Learning (MBRL)

Require: Policy/controller π_0, dynamics model p_θ, start state $\mathbf{x}_0$

1: **for** $i = 0, 1, \ldots$ **do**
2: Select π_i using exploration strategy, e.g. Equation (2.28)
3: Collect environment data set $\mathcal{D}_i$ using π_i; add to dataset $\mathcal{D}_{0:i} = \{\mathcal{D}_i \cup \mathcal{D}_{0:i-1}\}$
4: Update dynamics model p_θ using $\mathcal{D}_{0:i}$.
5: **end for**

ever, a central assumption of many methods is that both the system dynamics and cost function are *known a priori*. In contrast, we consider problems where the underlying dynamics modes $\{f_k\}_{k=1}^K$ and how the system switches between them α, are *not fully known a priori*. Further to this, the problem statement in Equation (2.4) contains a mode remaining constraint, which requires explicit knowledge of the desired dynamics mode and its state domain.

To apply control and planning techniques in systems with unknown dynamics, system identification emerged as a set of techniques for computing unknown parameters, e.g. mass of a component (Ljung, 1999). This is a two-staged approach which first learns about the environment and then uses this learned model to find the optimal controller. However, this approach learns about the environment globally and often incurs high costs during system identification.

Reinforcement Learning (RL) provides the most general framework for extending optimal control to problems with incomplete knowledge of the system dynamics. The classic text by Sutton and Barto, 2018 gives a general introduction to RL. The main goal is to learn good behaviours from interactions with an environment. Typically this is in the form of a state feedback controller, known as a policy, which makes an agent's interaction with the environment closed-loop. In contrast to the system identification approach, the goal of RL is to minimise costs (maximise rewards) during the learning process. Further to this, RL only needs to learn about the states relevant to solving the optimal control problem.

Model-based Reinforcement Learning

This thesis is interested in a subset of RL known as Model-Based Reinforcement Learning (MBRL). It solves the optimal control problem in Equation (2.2) by first learning a dynamics model and then using this learned model with model-based control techniques. As both Model-Free Reinforcement Learning (MFRL) and MBRL methods learn models, the model in the name refers to the dynamics model f, as MFRL approaches do not learn representations of the dynamics f. Algorithm 1 illustrates a common MBRL procedure where an agent incrementally explores its environment, collecting data $\mathcal{D}_i$ at each iteration i and updating its dynamics model p_θ.

As more systems are becoming data-driven, learning dynamics models for model-based control has shifted to needing task-centric methods that simultaneously learn about the environment, whilst optimising a controller to obtain low cumulative costs. The field of MBRL seeks to solve many optimal control problems by incrementally learning a dynamics model in this way (Chua et al., 2018; Deisenroth and Rasmussen, 2011). MBRL shares similarities with the system identification and control process, except that the dynamics learning and control are updated simultaneously. There are two main components to a MBRL algorithm: 1) a method for learning a dynamics model p_θ and 2) a controller (or policy) π that leverages the learned dynamics model p_θ. The controller may be a parameterised state feedback controller (policy) π, which is trained using MFRL algorithms and the learned dynamics model. Alternatively, the controller may take the form of a model-based control algorithm such as Model Predictive Control (MPC).

2.2.3 Model-based Control and Planning

This section reviews model-based control methods that leverage learned dynamics models in the MBRL setting. In the RL literature, model-based control is often

referred to as planning. The work in this thesis is primarily focused on model-based control techniques, in particular, trajectory optimisation.

Trajectory optimisation Instead of approximating a value function or a policy, it is possible to directly optimise the controls $\bar{\mathbf{u}} = \{\mathbf{u}_1, \dots, \mathbf{u}_{T-1}\}$ over a horizon T. This is known as trajectory optimisation and is given by,

$$\bar{\mathbf{u}} = \arg\min_{\bar{\mathbf{u}}} \mathbb{E}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \mid \mathbf{x}_0 = \mathbf{x}\right] \tag{2.6a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \epsilon, \tag{2.6b}$$

which finds an optimal sequence of controls $\bar{\mathbf{u}}$ for a given start state $\mathbf{x}_0$. Many recent works have used trajectory optimisation with learned dynamics models. For example, Nakka et al., 2021 developed a chance-constrained trajectory optimisation algorithm that leverages a dynamics model learned using a robust regression model. Rybkin et al., 2021 utilise a latent space dynamics model – learned with a convolutional neural network for the encoder and a recurrent neural network for the dynamics – which scales to high-dimensional inputs such as images. In MBRL, trajectory optimisers often exploit inaccuracies of the learned dynamics model. Boney et al., 2019 propose a trajectory optimisation approach that relives this exploitation by learning the dynamics using a denoising autoencoder. The resulting trajectory optimisation algorithm avoids regions of the learned dynamics model where no data has been observed.

Model predictive control Instead of directly applying the control inputs found with trajectory optimisation in an open-loop fashion, it is possible to obtain a closed-loop controller. This is achieved by iteratively applying the first control $\mathbf{u}_0$ and resolving the trajectory optimisation problem in Equation (2.6),

$$\pi_{\text{MPC}}(\mathbf{x}) = \arg\min_{\mathbf{u}_0} \min_{\bar{\mathbf{u}}} \mathbb{E}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \mid \mathbf{x}_0 = \mathbf{x}\right] \tag{2.7a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \epsilon, \tag{2.7b}$$

This is known as Model Predictive Control (MPC) (Eduardo F. and Carlos, 2007). However, in practice, it is often too computationally expensive to obtain real-time control with MPC. Many approximate solutions have been introduced in the literature, that seek to balance the computational complexity and accuracy trade-off differently (Betts, 1998).

For example, iterative Linear Quadratic Regulator (iLQR) can generate trajectories for nonlinear systems by iteratively approximating the dynamics to be linear around a nominal trajectory and optimising for the controls. iLQR works well for quadratic cost functions but can be used with any cost function by approximating the cost function with a second-order Taylor expansion. However, in this case, iLQR is susceptible to converging to terrible (local) optima if the true cost function is highly non-convex. Boedecker et al., 2014 present a real-time iLQR controller based on sparse GPs. Rohr et al., 2021 propose a novel LQR controller synthesis for linearised GP dynamics that yields robust controllers with respect to a probabilistic stability margin.

MPC has directly been used with ensembles of probabilistic neural networks (Chua et al., 2018; Nagabandi et al., 2020) and with GPs (Kamthe and Deisenroth, 2018). Lambert et al., 2019 control a quadcopter using online MPC and a dynamics model learned using probabilistic neural networks.

Offline trajectory optimisation Instead of solving the trajectory optimisation problem in Equation (2.6) online, it can be solved offline. For example, the state-control trajectory can be found offline and used as a reference trajectory for a tracking controller. Alternatively, the trajectory optimiser can be used offline to learn a state feedback controller (policy) using guided policy search (Levine and Koltun, 2013).

2.2.4 Constrained Control

This work aims to control multimodal dynamical systems subject to the mode remaining constraint in Equation (2.4). However, neither the underlying dynamics

modes nor how the system switches between them, are *known a priori*. To this end, this thesis is interested in model-based control techniques which can learn and enforce latent constraints.

It is common to require constraints on the states $\mathbf{x}$ and controls $\mathbf{u}$ of a controlled system. For example, an autonomous system may wish to remain in a subset of its state space where it knows its dynamics model is valid. The system may also be subject to constraints on the controls due to physical limitations, e.g. how quickly a quadcopter can accelerate and turn. Constraints of this type can be encoded via inequality constraints on the states $\mathbf{x}$ and controls $\mathbf{u}$,

$$c_{\mathbf{x}}(\mathbf{x}_t) \geq 0, \quad \forall t \geq 0, \tag{2.8}$$

$$c_{\mathbf{u}}(\mathbf{u}_t) \geq 0, \quad \forall t \geq 0. \tag{2.9}$$

The feasible regions of these constraints can be written as sets,

$$\mathcal{X}_{\text{feasible}} = \{\mathbf{x} \in \mathbb{R}^{D_x} \mid c_{\mathbf{x}}(\mathbf{x}) \geq 0\}, \tag{2.10}$$

$$\mathcal{U}_{\text{feasible}} = \{\mathbf{u} \in \mathbb{R}^{D_u} \mid c_{\mathbf{u}}(\mathbf{u}) \geq 0\}, \tag{2.11}$$

so the constraints can alternatively be written as,

$$\mathbf{x}_t \in \mathcal{X}_{\text{feasible}} \quad \forall t \geq 0 \tag{2.12}$$

$$\mathbf{u}_t \in \mathcal{U}_{\text{feasible}} \quad \forall t \geq 0. \tag{2.13}$$

For a parametric controller π, the control constraints can be encoded directly into the controller by parameterising it so that its range is restricted to $\mathcal{U}_{\text{feasible}}$, i.e. $\pi(\mathbf{x}_t) \in \mathcal{U}_{\text{feasible}}$, for all $\mathbf{x} \in \mathcal{X}_{\text{feasible}}$. The state constraints can be enforced by ensuring that the set is forward invariant.

Definition 2.2.1 (Forward invariant)**.** *Given a dynamical system* $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi(\mathbf{x}_t, t))$*, a set* $\mathcal{X}_{feasible}$ *is* ***forward invariant*** *under the controller* $\pi \in \Pi$*, iff, for all* $\mathbf{x}_0 \in \mathcal{X}_{feasible}$*, all future states remain in the set, i.e.* $f(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \in \mathcal{X}_{feasible} \quad \forall t$.

There are multiple approaches to enforcing state constraints via invariant sets. Two common approaches are Lyapunov functions (Lyapunov, 1992) and control barrier functions (Ames et al., 2019). Lyapunov functions are more restrictive than control barrier functions as they provide stability guarantees which are not a necessary condition to render $\mathcal{X}_{\text{feasible}}$ forward invariant. Although these are interesting directions for future work, they are out of the scope of this thesis.

Unknown constraints This work is interested in learning constraints whilst ensuring that they are satisfied. Ariafar et al., 2019; Gelbart et al., 2014 introduce algorithms to minimise an unknown objective (Bayesian optimisation) subject to unknown constraints. Sadigh and Kapoor, 2016 propose an MPC method that satisfies *a priori unknown* constraints with high probability. However, they do not deploy a strategy to actively learn about the constraints. In contrast, Schreiter et al., 2015 consider safe exploration for active learning. They distinguish safe and unsafe regions with a binary GP classifier, which is learned separately to the dynamics model. Their exploration strategy then considers the differential entropy of the dynamics GP and they use the GP classifier to define a set of safety constraints.

Stochastic constraints In stochastic systems it is not possible to make deterministic statements about constraints. This is because given a start state $\mathbf{x}_0$ the resulting trajectories are random variables. Therefore, the constraints $c_{\mathbf{x}}(\mathbf{x})$ are also random variables. To reason about constraints in stochastic systems, we need a method to measure uncertainty in this random variable.

A simple approach is to consider expected constraints $\mathbb{E}[c_{\mathbf{x}}(\mathbf{x})]$. However, although expected performance is a reasonable objective, expected constraints make less sense. For example, although the expected constraints $\mathbb{E}[c_{\mathbf{x}}(\mathbf{x})]$ hold, a system may still violate them frequently if the constraints variance $\mathbb{V}[c_{\mathbf{x}}(\mathbf{x})]$ is high. Ferber et al., 1958 proposed risk sensitivity which uses higher-order moments as well as the ex-

pected value. Value at risk (Duffie and Pan, 1997) is an even stronger notion, which guarantees constraint satisfaction with high probability.

MPC (Eduardo F. and Carlos, 2007) is the most direct method to embed constraints. At each time step, MPC ensures that the constraints hold over a given horizon. However, it is worth noting that these constraints still cannot be guaranteed in stochastic systems. For stochastic systems, Schwarm and Nikolaou, 1999 proposed to satisfy constraints with high probability. Such constraints are named chance constraints. Chance constraints are applicable in systems where the uncertainty arises from learning from observations. i.e. they are applicable with latent constraints.

2.3 Learning Dynamical Systems for Control

This section reviews methods for learning representations of dynamical systems for control. When learning representations of dynamical systems from observations it is important to consider the different forms of uncertainty. For example, when using a learned model for control, it is important to *know what we do not know*. This knowledge can be used to encode *risk-sensitive* control (avoid regions where the model cannot predict confidently) or to guide exploration. Section 2.4 discusses exploration strategies that leverage well-calibrated uncertainty estimates.

2.3.1 Sources of Uncertainty

This section characterises the uncertainty that arises in RL.

Aleatoric uncertainty Dynamical systems give rise to temporal observations arriving as a sequence $\bar{\mathbf{x}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. These measurements are often corrupted by (observation) noise due to imperfections in the measurement process. Even when it is known that there is uncertainty in the measurement process, there still remains uncertainty about its form. Given our current understanding of the real-world, many dynamical systems also appear to be inherently stochastic. This is due to our

inability to model certain phenomena accurately (e.g. turbulence). Stochasticity arising from state transitions in this way is known as process noise. Observation and process noise are the constituent sources of *aleatoric uncertainty*; uncertainties that are inherent in a system and cannot be reduced.

Epistemic uncertainty The uncertainty arising from learning the dynamics f from observations is known as *epistemic uncertainty*. For example, we may be uncertain about the structure of the model or the value of specific model parameters θ. This type of uncertainty accounts for those we could know, but we do *not know a priori*. As such, *epistemic uncertainty* can be reduced by exploring, collecting data and updating the model with this new experience.

Distinguishing these two sources of uncertainty is important in RL. For example, the *epistemic uncertainty* is useful for guiding exploration into regions of the system that have not been observed. In turn, this data can be used to reduce the model's *epistemic uncertainty* by updating the model. In contrast, driving the system into regions of the model with high *aleatoric uncertainty* is not desirable. Consider the case where the model is confident that the system is subject to high process noise in a particular region. Guiding the system into this region will not reduce the model's *epistemic uncertainty* because the model has already been trained on data from this region. Further to this, it may be undesirable to enter regions of high *aleatoric uncertainty* because they may result in poor performance or even catastrophic failure.

2.3.2 Learning Single-Step Dynamics Models

This work considers single-step dynamics models with the delta state formulation, which regularises the predictive distribution, given by $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon}$. Although multi-step dynamics models are an interesting direction for learning-based control, they are out of the scope of this thesis.

Single-step dynamics models have been deployed in a large variety of learning-based control algorithms. Early approaches include using single-step linear models (Schnei-

der, 1996) and single-step GP models (Deisenroth and Rasmussen, 2011; Hewing et al., 2020b; Koller et al., 2018; Rohr et al., 2021; Vinogradska et al., 2016) for low-dimensional control problems. More recently, single-step dynamics models have been learned using neural networks. For example, (Chua et al., 2018; Janner et al., 2019; Kurutach et al., 2018) use ensembles of neural networks and (Depeweg et al., 2017; Gal et al., 2016) use Bayesian neural networks with parametric uncertainty.

Probabilistic models *Mathematical models* are compact representations (sets of assumptions) that attempt to capture key features of the phenomenon of interest in a precise mathematical form. Probabilistic modelling provides the capability of constructing *mathematical models* that can represent and manipulate uncertainty in data, models, decisions and predictions. As such, linking observed data to underlying phenomena through probabilistic models is an interesting direction for modelling, analysing and controlling dynamical systems. It enables the uncertainty to be represented and manipulated; it provides a systematic way to combine observations with existing knowledge via a *mathematical model*. Learning representations of dynamical systems for control using probabilistic models has shown much promise. Moreover, learning single-step dynamics models that quantify uncertainty has been central to recent successes in MBRL (Chua et al., 2018; Janner et al., 2019).

Modelling a Bayesian belief over the dynamics function f provides a principled method for modelling *epistemic uncertainty* arising from learning from data, through the posterior distribution. GPs are the state-of-the-art approach for Bayesian non-parametric regression and are a building block for the methods presented in this thesis. GPs have become a popular choice for learning representations of dynamical systems (Buisson-Fenet et al., 2020; Nguyen-Tuong et al., 2009; Wang et al., 2018) and have been used for both RL (Deisenroth and Rasmussen, 2011; Doerr et al., 2017; Polymenakos et al., 2019; Vinogradska et al., 2020) and for MPC (Hewing et al., 2020a,b; Kamthe and Deisenroth, 2018; Koller et al., 2018).

2.3.3 Gaussian Processes

The mathematical machinery underpinning GPs is now detailed.

Multivariate Gaussian identities Inference techniques with GPs leverage multivariate Gaussian conditioning operations. As such, introducing the multivariate Gaussian identities is a natural place to start.

Gaussian distributions are popular in machine learning and control theory. This is not only due to their natural emergence in statistical scenarios (central limit theorem) but also their intuitiveness and mathematical properties that render their manipulation tractable and easy.

Consider a multivariate Gaussian whose random variables are partitioned into two vectors $\mathbf{f}$ and $\mathbf{u}$. The joint distribution takes the following form,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_\mathbf{f} \\ \boldsymbol{\mu}_\mathbf{u} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{ff}} & \boldsymbol{\Sigma}_{\mathbf{fu}} \\ \boldsymbol{\Sigma}_{\mathbf{uf}} & \boldsymbol{\Sigma}_{\mathbf{uu}} \end{bmatrix} \right),$$

where $\boldsymbol{\mu}_\mathbf{f}$ and $\boldsymbol{\mu}_\mathbf{u}$ represent the mean vectors, $\boldsymbol{\Sigma}_{\mathbf{ff}}$ and $\boldsymbol{\Sigma}_{\mathbf{uu}}$ represent the covariance matrices, and $\boldsymbol{\Sigma}_{\mathbf{uf}}$ and $\boldsymbol{\Sigma}_{\mathbf{fu}}$ represent the cross-covariance matrices. The marginalisation property of Gaussian distributions states that for two jointly Gaussian random variables, the marginals are also Gaussian,

$$p(\mathbf{f}) = \int p(\mathbf{f}, \mathbf{u}) \mathrm{d}\mathbf{u} = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}_\mathbf{f}, \boldsymbol{\Sigma}_{\mathbf{ff}}), \tag{2.14}$$

$$p(\mathbf{u}) = \int p(\mathbf{f}, \mathbf{u}) \mathrm{d}\mathbf{f} = \mathcal{N}(\mathbf{u} \mid \boldsymbol{\mu}_\mathbf{u}, \boldsymbol{\Sigma}_{\mathbf{uu}}). \tag{2.15}$$

Conveniently, the conditional densities are also Gaussian,

$$\mathbf{f} \mid \mathbf{u} \sim \mathcal{N}\big(\boldsymbol{\mu}_\mathbf{f} + \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}(\mathbf{u} - \boldsymbol{\mu}_\mathbf{u}), \boldsymbol{\Sigma}_{\mathbf{ff}} - \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}\boldsymbol{\Sigma}_{\mathbf{uf}}\big), \tag{2.16}$$

$$\mathbf{u} \mid \mathbf{f} \sim \mathcal{N}\big(\boldsymbol{\mu}_\mathbf{u} + \boldsymbol{\Sigma}_{\mathbf{uf}}\boldsymbol{\Sigma}_{\mathbf{ff}}^{-1}(\mathbf{f} - \boldsymbol{\mu}_\mathbf{f}), \boldsymbol{\Sigma}_{\mathbf{uu}} - \boldsymbol{\Sigma}_{\mathbf{uf}}\boldsymbol{\Sigma}_{\mathbf{ff}}^{-1}\boldsymbol{\Sigma}_{\mathbf{fu}}\big). \tag{2.17}$$

Consider the case where $\mathbf{u}$ represents some observations and $\mathbf{f}$ represents a new test location. Equation (2.16) can be used to make inferences in the location $\mathbf{f}$ given the observations $\mathbf{u}$, i.e. make sophisticated interpolations on the measurements based on their closeness. In real-world scenarios, it is desirable to consider the entire input domain, instead of simply pre-selecting a discrete set of locations. Gaussian processes provide this mathematical machinery.

Gaussian processes Informally, GPs are a generalisation of the multivariate Gaussian distribution, indexed by an input domain as opposed to an index set. Similar to how a sample from an $N-$dimensional multivariate Gaussian is an $N-$dimensional vector, a sample from a GP is a random function over its domain. Formally, a GP is defined as follows,

Definition 2.3.1 (Gaussian process)**.** *A Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution (Rasmussen and Williams, 2006).*

More intuitively, a Gaussian process is a distribution over functions $f(\cdot) : \mathcal{X} \to \mathbb{R}$ defined over an input domain $\mathcal{X} \in \mathbb{R}^{D_f}$. Whilst Gaussian distributions represent distributions over finite-dimensional vectors, GPs represent distributions over infinite-dimensional functions. A GP is fully defined by a mean function $\mu(\cdot) : \mathcal{X} \to \mathbb{R}$ and a covariance function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (also known as a kernel),

$$f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot, \cdot)), \tag{2.18}$$

$$\mu(\cdot) = \mathbb{E}[f(\cdot)], \tag{2.19}$$

$$k(\cdot, \cdot') = \mathbb{E}[(f(\cdot) - \mu(\cdot))(f(\cdot') - \mu(\cdot'))]. \tag{2.20}$$

Importantly, for a given set of training inputs from the functions domain $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the associated function values $\mathbf{f} = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)\}$, are jointly Gaussian. This results in an $N-$dimensional multivariate Gaussian random variable $\mathbf{f}$, given by,

$$\mathbf{f} \sim \mathcal{N}(\mu(\mathbf{X}), k(\mathbf{X}, \mathbf{X})). \tag{2.21}$$

A common kernel that is used throughout this thesis is the Squared Exponential kernel with Automatic Relevance Determination (ARD), given by,

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D_f}\left(\frac{x_d - x_i'}{l_d}\right)\right), \tag{2.22}$$

where σ_f^2 represent the signal variance and l_d is a lengthscale parameter associated with input dimension d. The lengthscale parameter determines the length of the "wiggles" in the function and the signal variance σ_f^2 determines the average deviation of the function from its mean.

Given mean and kernel functions with parameters $\boldsymbol{\theta}$, the marginal distribution is given by,

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\mathbf{f} \mid \mu(\mathbf{X}), k(\mathbf{X}, \mathbf{X})). \tag{2.23}$$

where the dependency on the parameters $\boldsymbol{\theta}$ has been dropped, i.e. $p(\mathbf{f} \mid \mathbf{X}) = p(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\theta})$. This simplification will be used throughout this thesis for notational conciseness. By definition, these observations $\mathbf{f}$ are jointly Gaussian with any unobserved function value $f_* = f(\mathbf{x}_*)$ at a new test input,

$$\begin{bmatrix} \mathbf{f} \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{X}) \\ \mu(\mathbf{x}_*) \end{bmatrix} \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) & k(\mathbf{X}, \mathbf{x}_*) \\ k(\mathbf{x}_*, \mathbf{X}) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right).$$

Given the multivariate Gaussian conditionals in Equation (2.16), it is easy to see how the distribution over the test function value f_*, can be obtained by conditioning on the observations,

$$\begin{aligned} p(f_* \mid \mathbf{x}_*, \mathbf{f}, \mathbf{X}) &= \mathcal{N}(f_* \mid \mu, \sigma^2) \\ \mu &= \mu(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{X})k(\mathbf{X}, \mathbf{X})^{-1}(\mathbf{f} - \mu(\mathbf{X})) \\ \sigma^2 &= k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})k(\mathbf{X}, \mathbf{X})^{-1}k(\mathbf{X}, \mathbf{x}_*). \end{aligned} \tag{2.24}$$

It is typical in real-world modelling scenarios that observations of the true function values $\mathbf{f}$ are not directly accessible. Instead, observations are usually corrupted by noise,

$$\mathbf{y} = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}\big(\mathbf{0}, \sigma_n^2 \mathbf{I}\big). \tag{2.25}$$

where σ_n^2 is the noise variance. In this scenario, the function values $\mathbf{f}$ become latent variables and a Gaussian likelihood is introduced,

$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}\big(\mathbf{y} \mid \mathbf{f}, \sigma_n^2 \mathbf{I}\big), \tag{2.26}$$

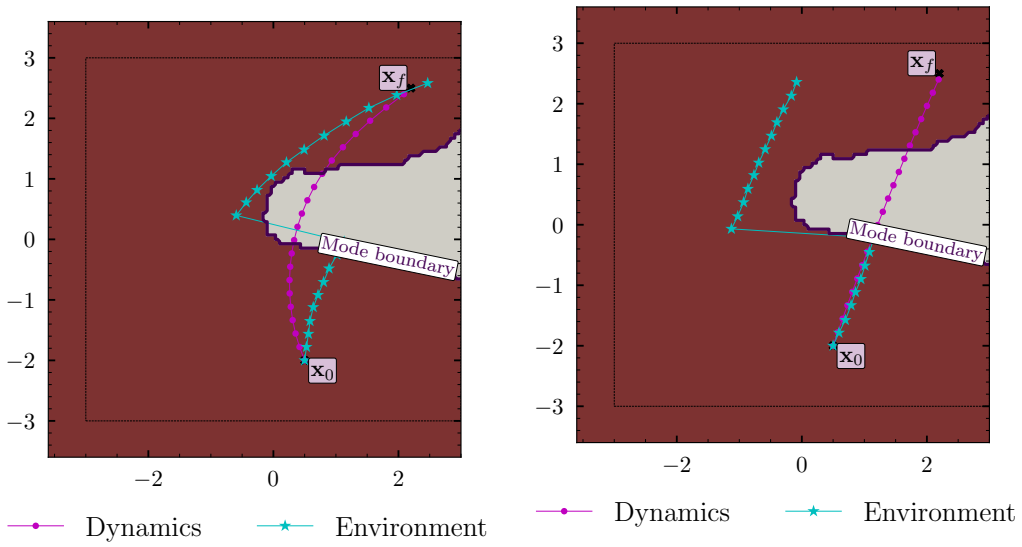
to relate the observations to the latent function values $\mathbf{f}$. The predictive distribution for a test input $\mathbf{x}_*$ follows from Equation (2.16),

$$\begin{aligned} p(f_* \mid \mathbf{x}_*, \mathbf{y}, \mathbf{X}) &= \mathcal{N}(f_* \mid \mu, \sigma^2) \\ \mu &= \mu(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{X})\big(k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\big)^{-1}(\mathbf{y} - \mu(\mathbf{X})) \\ \sigma^2 &= k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})\big(k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})\big)^{-1} k(\mathbf{X}, \mathbf{x}_*). \end{aligned} \tag{2.27}$$

This predictive distribution is the GP posterior. Importantly, the predictive variance σ^2 in Equation (2.27) quantifies the *epistemic uncertainty* associated with making a prediction at the test input $\mathbf{x}_*$, whilst the value of the noise variance σ_n^2 quantifies the *aleatoric uncertainty*. This structured handling of uncertainty makes GPs extremely powerful for learning representations of dynamical systems.

2.3.4 Learning Multimodal Dynamical Systems

In contrast to the previously presented approaches, we are interested in learning representations of multimodal dynamical systems. Figure 2.2a demonstrates the shortcomings of learning a GP dynamics model for the quadcopter navigation problem in the illustrative example from Section 1.1. It shows the results of performing trajectory optimisation in a GP dynamics model trained on state transitions sampled



*(a) GP dynamics model trained on state transitions sampled from **both** dynamics modes.*

*(b) GP dynamics model trained on state transitions sampled from **only the desired** (operable) dynamics mode.*

Figure 2.2: *Trajectory optimisation results obtained using a GP dynamics model trained on different data sets. The optimised control trajectories are rolled out in the GP dynamics model (magenta) and in the environment (cyan). The state trajectories are overlayed on the gating mask, which indicates where each mode governs the dynamics. White indicates the turbulent dynamics mode and red indicates the desired (operable) mode.*

from **both** dynamics modes. The GP has not been able to learn a representation of the dynamics which is true to the underlying system, due to the discontinuities associated with the multimodal transition dynamics (changing lengthscales/noise variances etc). The trajectory optimiser was able to find a trajectory from the start state $\mathbf{x}_0$, to the target state $\mathbf{x}_f$ in the GP dynamics (magenta). However, as the GP dynamics do not accurately represent the true underlying dynamics, the state trajectory resulting from rolling out the optimised controls in the environment (cyan) does not match the GP dynamics trajectory (magenta). This example motivates the need to correctly identify the underlying modes when learning representations of multimodal dynamical systems for control.

Figure 2.2b shows results after training on state transitions from only the desired, operable dynamics mode (red). The learned dynamics model can accurately predict state transitions in the desired dynamics mode (red). However, as this approach only

considers the dynamics of the desired mode, trajectories in the environment (cyan) deviate from those planned in the learned dynamics model (magenta) when they pass through the turbulent mode. This is problematic because the trajectory passes through the turbulent dynamics mode (which may lead to catastrophic failure) and does not reach the target state $\mathbf{x}_f$. Without inferring information regarding how the system switches between its underlying dynamics modes, it is not possible to encode mode remaining behaviour into control algorithms.

Remark. *As the underlying dynamics modes and how the system switches between them, are not fully known a priori, partitioning the data set and learning the dynamics model in Figure 2.2b is not possible in realistic scenarios.*

Following standard methodologies, the trajectories in Figure 2.2 were found by minimising the expected cost under the state distribution, resulting from cascading single-step predictions through the GP dynamics model using the moment matching approximation (Kuss, 2006). A terminal state cost term favoured trajectories ending at the target state and a quadratic integral control cost term regularised the controls to encode the notion of "minimal effort" control.

Methods for learning probabilistic multimodal dynamics have been proposed. Moerland et al., 2017b use deep generative models, namely a conditional Variational Auto-Encoder (VAE), to learn multimodal transition dynamics for MBRL. Kaiser et al., 2020 use the data association with GPs model; a Bayesian model that learns independent dynamics modes whilst maintaining a probabilistic belief over which mode is responsible for predicting at a given input location. McKinnon and Schoellig, 2017 also use an approach based on GPs, except that they use a Mixtures of Gaussian Process Experts (MoGPE) method to learn the switching behaviour for robot dynamics online.

Latent spaces for control It is worth noting that the introduction of *latent variables* into probabilistic models is a key component providing them with interesting

and powerful capabilities for synergising model learning and control. For example, Hafner et al., 2019; Rybkin et al., 2021 learn *latent spaces* which provide convenient spaces for control (or planning). Figure 2.2b highlights the need for learning informative latent variables representing how the system switches between the underlying dynamics modes. Without such information, it is not possible to encode the notion of mode remaining/avoiding behaviour. As such, this work is interested in learning *latent spaces* that are rich with information regarding how a system switches between its underlying dynamics modes.

2.4 Uncertainty-based Exploration Strategies

Reinforcement Learning (RL) agents face a trade-off between *exploration*, where they seek to explore the environment and improve their models, and *exploitation*, where they make decisions which are optimal for the data observed so far. There are many approaches from the literature used to tackle the exploration-exploitation trade-off. In MBRL, the goal is often to reduce the real-world sample complexity at the cost of increased model sample complexity.

There are two main uncertainties which are often modelled and used for exploration. Value-based methods base their exploration on the uncertainty of the value function V at the current state $\mathbf{x}_t$. Actions with higher value estimates get higher probabilities of being selected based on uncertainty estimates around the values (Auer, 2002; Moerland et al., 2017a). An alternative approach for MBRL is to use state-based exploration. This approach is state-specific, reward independent and usually seeks states with high *epistemic uncertainty*. This section recaps some relevant exploration strategies that fit into the general MBRL procedure shown in Algorithm 1.

Greedy exploitation One of the most commonly used exploration strategies is to select the controller that maximises the expected performance under the learned

dynamics model $p_\theta(f \mid \mathcal{D}_{0:i-1})$. Note that i denotes the iteration/episode number of the MBRL loop. This greedy strategy is given by,

$$\pi_i^{\text{greedy}} = \arg\max_{\pi \in \Pi} \mathbb{E}_{f \sim p_\theta(f \mid \mathcal{D}_{0:i-1})}[-J(f, \pi)]. \tag{2.28}$$

Note that the negative sign is used because the objective $J(f, \pi)$ is based on a cost function and not a reward function. This approach is used in PILCO (Deisenroth and Rasmussen, 2011) and GP-MPC (Kamthe and Deisenroth, 2018) where the dynamics are represented using GPs and the moment matching approximation is used to cascade single-step predictions. Parmas et al., 2018 propose PIPPS, a similar approach to PILCO, except that they use Monte Carlo methods to propagate uncertainty forward in time, instead of using the moment matching approximation. Similarly, PETS (Chua et al., 2018) uses this exploration strategy but represents the dynamics using ensembles of probabilistic neural networks. This strategy initially favours exploring regions of the environment where the learned dynamics model is not confident, i.e. has high *epistemic uncertainty*. Once it has gathered knowledge of the environment and the model's epistemic uncertainty has been reduced, it favours maximising the objective function $J(f, \pi)$.

Thompson sampling An alternative and theoretically grounded strategy is Thompson sampling. This approach samples a single model $f_i \sim p_\theta(f \mid \mathcal{D}_{0:i})$ at every iteration i and uses the sampled model to optimise the controller. This is given by,

$$\pi_i^{\text{thompson}} = \arg\max_{\pi \in \Pi} -J(f_i, \pi) \quad \text{s.t.} \quad f_i \sim p_\theta(f \mid \mathcal{D}_{0:i}). \tag{2.29}$$

In general, it is intractable to sample from $p_\theta(f \mid \mathcal{D}_{0:i})$. Note that after the sampling step this problem is equivalent to greedy exploitation.

Alternatively, some MBRL algorithms, such as Sekar et al., 2020, adopt a two-phase exploration strategy. The first phase is interested in exploring the environment and summarising this past experience in the form of a model. The second phase then

seeks to solve a downstream task, for which it is given a cost (reward) function. This two-stage approach does not require an objective that changes its exploration-exploitation balance as it gathers more knowledge of the environment.

Active learning Active learning is a class of exploration algorithms which fit into the two-phase exploration approach. The goal of information-theoretic active learning is to reduce the number of possible hypotheses as fast as possible, e.g. minimise the uncertainty associated with the parameters using Shannon's entropy (Cover and Joy, 2006),

Differential Entropy Let X be a continuous random variable, with a probability density $p(X)$, whose support is a set $\mathcal{X}$. The differential entropy $H[X]$ is then defined as,

$$H[X] = -\int_{\mathcal{X}} p(X)\log p(X)\mathrm{d}X. \tag{2.30}$$

In contrast to greedy exploitation, active learning does not seek to maximise a black-box objective. Instead, it is only interested in exploration. There are many approaches to active learning in the static setting, i.e. in systems where an arbitrary state $\mathbf{x}$ can be sampled. In contrast, dynamical systems must be steered to $\mathbf{x}$ through the unknown dynamics f through a sequence of controls $\bar{\mathbf{u}}$. Thus, information gain along the trajectory must also be considered. As highlighted by Buisson-Fenet et al., 2020, information gain in dynamical systems is fundamentally different to the static problem addressed by Krause et al., 2008 and Houlsby et al., 2011. The goal is to pick the most informative control trajectory $\bar{\mathbf{u}}$ whilst observing $\bar{\mathbf{x}}$.

Recent work has addressed active learning in GP dynamics models. Schreiter et al., 2015 propose a greedy entropy-based strategy that considers the entropy of the next state. Buisson-Fenet et al., 2020 also propose a greedy entropy-based strategy except that they consider the entropy accumulated over a trajectory. In contrast, Capone et al., 2020; Yu et al., 2021 propose using the mutual information.

Mutual Information Given two sets of random variables, $\mathbf{X}$ and $\mathbf{F}$, with joint density $p(\mathbf{X}, \mathbf{F})$ the mutual information (Cover and Joy, 2006) is given by,

$$I[\mathbf{X}; \mathbf{F}] = \int p(\mathbf{X}, \mathbf{F}) \log \frac{p(\mathbf{X}, \mathbf{F})}{p(\mathbf{X})p(\mathbf{F})} \mathrm{d}X \mathrm{d}F, \tag{2.31}$$

and its well-known relationship to differential entropy $H[\cdot]$ is given by,

$$I[\mathbf{X}; \mathbf{F}] = H[\mathbf{X}] - H[\mathbf{X} \mid \mathbf{F}]. \tag{2.32}$$

Capone et al., 2020 find the most informative state as the one that minimises the mutual information between it and a set of reference states (a discretisation of the domain). They then find a set of controls to drive the system to this most informative state. Given a fixed number of time steps, their method yields a better model than the greedy entropy-based strategies. Yu et al., 2021 propose an alternative approach that leverages their Gaussian Process State Space Model (GPSSM) inference scheme to estimate the mutual information between all the variables in time $I[\mathbf{y}_{1:t}, \hat{\mathbf{y}}_{t+1}; \mathbf{f}_{1:t+1}]$. Here $\mathbf{y}_{1:t}$ denotes the set of observed outputs and $\hat{\mathbf{y}}_{t+1}$ denotes the output predicted by the GPSSM. This contrasts with other approaches, which study the latest mutual information $I[\hat{\mathbf{y}}_{t+1}; \mathbf{f}_{t+1}]$.

Myopic active learning In RL and control it is standard to consider objectives over a potentially infinite horizon. However, active learning objectives often myopically consider the information gain at the next query point only. In contrast, it is possible to consider the information gain over a potentially infinite horizon, reliving this myopia. The mutual information approaches in Capone et al., 2020; Yu et al., 2021 fall into this myopic category as they only maximise the information gain at the next time step. In contrast, the entropy-based strategy in Buisson-Fenet et al., 2020 considers the entropy over a horizon.

3 Probabilistic Inference for Learning Multimodal Dynamical Systems

> All models are wrong, but some are useful.
>
> *George Box*

This chapter is concerned with *learning* representations of multimodal dynamical systems for model-based control. It is interested in systems where both the underlying dynamics modes and how the system switches between them, are *not fully known a priori*. This chapter assumes access to a data set of state transitions $\mathcal{D}$, previously sampled from the system at a constant frequency, i.e. with a fixed time-step.

Following the motivation in Section 2.3.4, this chapter seeks to identify the underlying dynamics modes correctly whilst inferring latent structure that can be exploited for control. The main goals of this chapter can be summarised as follows,

1. accurately *identify* the true underlying dynamics modes,
2. learn *latent spaces* for planning/control.

The probabilistic model constructed in this chapter resembles a Mixtures of Gaussian Process Experts (MoGPE) with a GP-based gating network and is named Mixtures of Sparse Variational Gaussian Process Experts (MoSVGPE). Following other MoGPE methods, it is evaluated on the motorcycle data set (Silverman, 1985). It is

then tested on a real-world quadcopter data set representing the illustrative example detailed in Section 1.1.

3.1 Problem Statement

This work considers learning representations of *unknown* or *partially unknown*, stochastic, multimodal, nonlinear dynamical systems. That is, it seeks to learn a representation of the dynamics from the problem statement in Section 2.1. This chapter considers single-step dynamics models with the delta state formulation that regularises the predictive distribution. The dynamics are given by,

$$\begin{aligned} \Delta \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon} && \text{(3.1a)} \\ &= f_k(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon}_k \quad \text{if} \quad \alpha(\mathbf{x}_t) = k \quad \forall k \in \{1, \dots K\} \\ \boldsymbol{\epsilon}_k &\sim \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\epsilon}_k}), && \text{(3.1b)} \end{aligned}$$

with continuous states $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^{D_x}$, continuous controls $\mathbf{u}_t \in \mathcal{U} \subseteq \mathbb{R}^{D_u}$ and time index t. The state difference between time t and $t+1$ is denoted $\Delta \mathbf{x}_{t+1} = \mathbf{x}_{t+1} - \mathbf{x}_t$. At any given time step t, one of the K dynamics modes $\{f_k : \mathcal{X}_k \times \mathcal{U} \to \mathcal{X}\}_{k=1}^K$ and associated noise models $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_k})$, are selected by a discrete mode indicator variable $\alpha_t \in \mathcal{A} = \{1, \dots, K\}$. This work assumes that the discrete mode indicator variable depends on the state $\mathbf{x}$ such that it is governed by a discrete mode indicator function $\alpha : \mathcal{X} \to \mathcal{A}$.

This chapter assumes access to historical data comprising state transitions from E trajectories of length T, sampled with a fixed time step $\Delta t = t_*$. The data set has $N = ET$ elements and we abuse notation by appending the independent trajectories along time to get the data set $\mathcal{D} = \{(\mathbf{x}_t, \mathbf{u}_t), \Delta \mathbf{x}_{t+1}\}_{t=0}^{N-1}$.

To ease notation, our modelling only considers a single output dimension. The extension to multiple output dimensions follows from standard GP methodologies and is detailed where necessary. To further ease notation, the state-control input

domain is denoted $\mathcal{Z} = \mathcal{X} \times \mathcal{U} \subseteq \mathbb{R}^D$ and a single state-control input is denoted $\mathbf{x}_n = (\mathbf{x}_t, \mathbf{u}_t)$. Given this formulation, this chapter aims to learn the mapping f, which switches between K different functions f_k. This can be cast as a regression problem,

$$\underbrace{\Delta \mathbf{x}_{t+1}}_{y_n} = \underbrace{f_k(\mathbf{x}_t, \mathbf{u}_t)}_{f_k\,(\mathbf{x}_n)} + \epsilon_k \quad \text{if} \quad \alpha_n = k, \tag{3.2}$$

where both the latent dynamics functions $\{f_k\}_{k=1}^K$ and how the system switches between them α, must be inferred from observations. A single observation is denoted as $(\mathbf{x}_n, y_n) = ((\mathbf{x}_t, \mathbf{u}_t), \Delta \mathbf{x}_{t+1})$. The set of all inputs is denoted as $\mathbf{X} \in \mathbb{R}^{N \times D}$ and the set of all outputs as $\mathbf{y} \in \mathbb{R}^{N \times 1}$. With this notation, the regression problem can be written as,

$$y_n = f_k(\mathbf{x}_n) + \epsilon_k \quad \text{if} \quad \alpha_n = k. \tag{3.3}$$

3.2 Preliminaries

Gaussian Processes (GPs) are the state-of-the-art approach for Bayesian nonparametric regression and they provide a powerful mechanism for encoding expert domain knowledge. They are flexible enough to model arbitrary smooth functions with the simplicity of only requiring inference over a small number of interpretable parameters, such as lengthscales and the contributions of signal and noise variance in the data. These properties are induced by the *covariance function*, which models the covariance between observations. As such, MoGPE methods are a promising direction for modelling multimodal dynamical systems. This section recaps the MoGPE concepts that this chapter builds upon.

Mixture Models Mixture models are a natural choice for modelling multimodal systems. Given an input $\mathbf{x}_n$ and an output y_n, mixture models model a mixture of distributions over the output,

$$p(y_n \mid \mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{k=1}^{K} \underbrace{\Pr(\alpha_n = k \mid \boldsymbol{\phi})}_{\text{mixing probability}} \underbrace{p(y_n \mid \alpha_n = k, \mathbf{x}_n, \boldsymbol{\theta}_k)}_{\text{component } k}. \tag{3.4}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ represent model parameters and α_n is a discrete indicator variable assigning observations to components. The predictive distribution $p(y_n \mid \mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\phi})$ consists of K mixture components $p(y_n \mid \alpha_n = k, \mathbf{x}_n, \boldsymbol{\theta}_k)$ that are weighted according to their mixing probabilities $\Pr(\alpha_n = k \mid \boldsymbol{\phi})$.

Mixture of Experts The Mixture of Experts (MoE) model (Jacobs et al., 1991) is an extension where the mixing probabilities depend on the input variable $\Pr(\alpha_n = k \mid \mathbf{x}_n, \boldsymbol{\phi})$, and are collectively referred to as the *gating network*. The individual component densities $p(y_n \mid \alpha_n = k, \mathbf{x}_n, \boldsymbol{\theta}_k)$ are then referred to as *experts*, as at different regions in the input space, different components are responsible for predicting. Given K experts $p(y_n \mid \alpha_n = k, \mathbf{x}_n, \boldsymbol{\theta}_k)$, each with parameters $\boldsymbol{\theta}_k$, the MoE marginal likelihood is given by,

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \underbrace{\Pr(\alpha_n = k \mid \mathbf{x}_n, \boldsymbol{\phi})}_{\text{gating network}} \underbrace{p(y_n \mid \alpha_n = k, \mathbf{x}_n, \boldsymbol{\theta}_k)}_{\text{expert } k}, \tag{3.5}$$

where α_n is the expert indicator variable assigning observations to experts. The probability mass function over the expert indicator variable is referred to as the gating network and indicates which expert governs the model at a given input location. See Yuksel et al., 2012 for a survey of MoE methods. Note the correspondence between the expert indicator variable α_n and the mode indicator variable from Equation (3.1).

Nonparametric Mixtures of Experts Modelling the experts as GPs gives rise to a class of powerful models known as Mixtures of Gaussian Process Experts

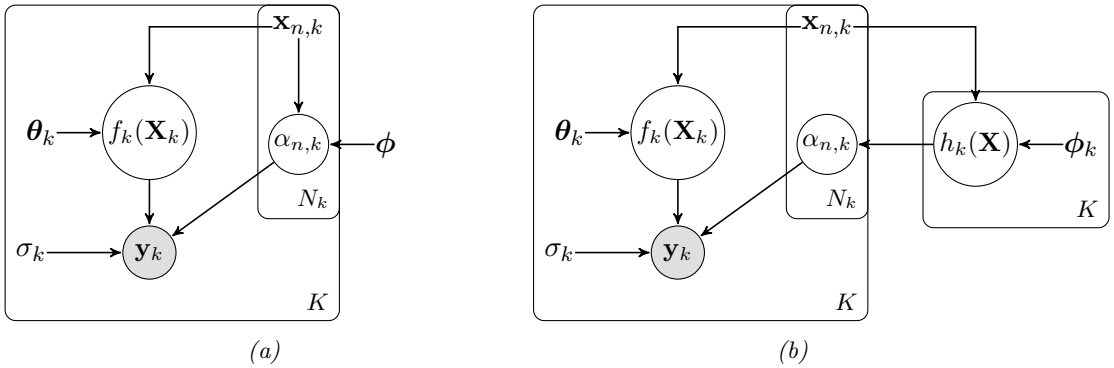


Figure 3.1: *Graphical models where the outputs* $\mathbf{y} = \{\mathbf{y}_k\}_{k=1}^K$ *are generated by mapping the inputs* $\mathbf{X} = \{\mathbf{X}_k\}_{k=1}^K$ *through the latent processes. An input assigned to expert* k *is denoted* $\mathbf{x}_{n,k}$ *and the sets of all* N_k *inputs and outputs assigned to expert* k *are denoted* $\mathbf{X}_k$ *and* $\mathbf{y}_k$ *respectively. The experts are shown on the left of each model and the gating network on the right. The generative processes involve evaluating the gating network and sampling an expert mode indicator variable* α_n*. The indicated mode's latent function* f_k *and noise model* $\mathcal{N}(0, \sigma_k)$ *are then evaluated to generate the output* y_n*. (a) shows the Mixture of Gaussian Process Experts model first presented in Rasmussen and Ghahramani, 2001 but without the Dirichlet process prior on the gating network. This represents the basic conditional model, not the full generative model over both the inputs and outputs as presented in Meeds and Osindero, 2006. (b) shows our model with a GP-based gating network which involves evaluating* K *latent gating functions* $\mathbf{h}$ *and normalising their output to obtain the mixing probabilities* $\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n))$*. The mode indicator variable* α_n *is then sampled from the Categorical distribution governed by these probabilities.*

(MoGPE). They can model multimodal distributions as they model a mixture of distributions over the outputs, usually a Gaussian mixture in the regression setting (Rasmussen and Ghahramani, 2001; Tresp, 2000). They can model non-stationary functions as each expert learns separate hyperparameters (lengthscales, noise variances etc). Many MoGPE methods have been proposed, and in general, they differ in the formulation of their gating network and their approximate inference algorithms.

Rasmussen and Ghahramani, 2001 highlighted that the traditional MoE marginal likelihood does not apply when the experts are nonparametric. This is because the model assumes that the observations are i.i.d. given the model parameters, which is contrary to GP models, which model the dependencies in the joint distribution, given the hyperparameters. Rasmussen and Ghahramani, 2001 point out that there is a joint distribution corresponding to every possible combination of assignments (of

observations to experts). The marginal likelihood is then a sum over exponentially many (K^N) sets of assignments,

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\phi}) &= \sum_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha} \mid \mathbf{X}, \boldsymbol{\phi}) p(\mathbf{y} \mid \boldsymbol{\alpha}, \mathbf{X}, \boldsymbol{\theta}) \\ &= \sum_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha} \mid \mathbf{X}, \boldsymbol{\phi}) \left[\prod_{k=1}^{K} p(\mathbf{y}_k \mid \mathbf{X}_k, \boldsymbol{\theta}_k) \right], \end{aligned} \tag{3.6}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_N\}$ represents a set of assignments for all observations. Figure 3.1a shows its graphical model representation. Note that $\mathbf{X}_k = \{\mathbf{x}_n : \alpha_n = k\}_{n=1}^{N_k}$ and $\mathbf{y}_k = \{y_n : \alpha_n = k\}_{n=1}^{N_k}$ represent the N_k inputs and outputs assigned to the k^{th} expert respectively. This distribution factors into the product over experts, where each expert models the joint Gaussian distribution over the observations assigned to it. Assuming a mixture of Gaussian process regression models, the marginal likelihood in Equation (3.6) can be expanded to show each of the experts' latent variables,

$$p(\mathbf{y} \mid \mathbf{X}) = \sum_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha} \mid \mathbf{X}, \boldsymbol{\phi}) \left[\prod_{k=1}^{K} \mathbb{E}_{p(f_k(\mathbf{X}_k))} \left[\prod_{n=1}^{N_k} p(y_n \mid f_k(\mathbf{x}_n)) \right] \right], \tag{3.7}$$

where each expert follows the standard Gaussian likelihood model,

$$y_n = f_k(\mathbf{x}_n) + \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, \sigma_k^2), \tag{3.8}$$

$$p(y_n \mid f_k(\mathbf{x}_n)) = \mathcal{N}\big(y_n \mid f_k(\mathbf{x}_n), \sigma_k^2\big), \tag{3.9}$$

with f_k and σ_k^2 representing the latent function and the noise variance associated with the k^{th} expert. Note that for notational conciseness the dependency on $\alpha_n = k$ is dropped from $p(y_n \mid f_k(\mathbf{x}_n))$ as it is implied by the mode indexing f_k. The dependence on $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ is also dropped from here on in. Independent GP priors are placed on each of the expert's latent functions,

$$f_k(\mathbf{X}_k) \sim \mathcal{N}(\mu_k(\mathbf{X}_k), k_k(\mathbf{X}_k, \mathbf{X}_k)) \tag{3.10}$$

where $\mu_k(\cdot)$ and $k_k(\cdot,\cdot)$ represent the mean and covariance functions associated with the k^{th} expert respectively. This leads to each expert resembling a standard GP regression model with a Gaussian likelihood. For notational conciseness the dependence on the inputs $\mathbf{X}_k$ and hyperparameters $\boldsymbol{\theta}_k$ is dropped for each GP prior, i.e. $p(f_k(\mathbf{X}_k)) = p(f_k(\mathbf{X}_k) \mid \mathbf{X}_k, \boldsymbol{\theta}_k)$.

3.3 Identifiable Mixtures of Gaussian Process Experts

Motivated by improving identifiability and learning latent spaces for control, this work adopts a GP-based gating network resembling a GP classification model, similar to that used in the original MoGPE model (Tresp, 2000). The GP-based gating network can be used to constrain the set of admissible functions through the placement of informative GP priors on the gating functions. Further to this, in Chapter 4, the geometry of the GP-based gating network is used to encode mode remaining behaviour into control strategies. Chapter 6 then leverages the power of the GP-based gating network to construct an explorative trajectory optimisation algorithm that can consider the information gain over trajectories, as opposed to just at the next state.

The marginal likelihood is given by,

$$p(\mathbf{y} \mid \mathbf{X}) = \sum_{\boldsymbol{\alpha}} \underbrace{\mathbb{E}_{p(\mathbf{h}(\mathbf{X}))}\left[\prod_{n=1}^{N} P(\alpha_n \mid \mathbf{h}(\mathbf{x}_n))\right]}_{\text{GP gating network}} \underbrace{\prod_{k=1}^{K} p(\{y_n : \alpha_n = k\} \mid \{\mathbf{x}_n : \alpha_n = k\})}_{\text{experts}} \tag{3.11}$$

where the gating network resembles a GP classification model, with a factorised classification likelihood $P(\alpha_n \mid \mathbf{h}(\mathbf{x}_n))$ dependent on input dependent functions $\mathbf{h} =$

$\{h_k : \mathcal{Z} \to \mathbb{R}\}_{k=1}^{K}$, known as gating functions. The probability mass function over the expert indicator variable is given by,

$$P(\alpha_n \mid \mathbf{h}(\mathbf{x}_n)) = \prod_{k=1}^{K} (\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n)))^{[\alpha_n = k]}, \tag{3.12}$$

where $[\alpha_n = k]$ denotes the Iverson bracket. The probabilities of this Categorical distribution $\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n))$ are governed by a classification likelihood (Bernoulli/-Softmax). Fig. 3.1b shows the graphical model where the K latent gating functions $\mathbf{h}$ are evaluated and normalised to obtain the mixing probabilities $\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n))$. The mode indicator variable α_n is then sampled from the Categorical distribution governed by these probabilities. The indicated expert's latent function f_k and noise model $\mathcal{N}(\mathbf{0}, \sigma_k^2)$ are then evaluated to generate the output y_n.

Softmax ($K > 2$) In the general case, when there are more than two experts, the gating network's likelihood is defined as the Softmax function,

$$\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n)) = \text{softmax}_k(\mathbf{h}(\mathbf{x}_n)) = \frac{\exp(h_k(\mathbf{x}_n))}{\sum_{j=1}^{K} \exp(h_j(\mathbf{x}_n))}. \tag{3.13}$$

Each gating function h_k describes how its corresponding mode's mixing probability varies over the input space. Modelling the gating network with input-dependent functions enables informative prior knowledge to be encoded through the placement of GP priors on each gating function. Further to this, if the modes are believed to only vary over a subset of the state-control input space, then the gating functions can depend only on this subset. Independent GP priors are placed on each gating function, giving the gating network prior,

$$\mathbf{h}(\mathbf{X}) \sim \prod_{k=1}^{K} \mathcal{N}\Big(\hat{\mu}_k(\mathbf{X}), \hat{k}_k(\mathbf{X}, \mathbf{X})\Big) \tag{3.14}$$

where $\hat{\mu}_k(\cdot)$ and $\hat{k}_k(\cdot, \cdot)$ are the mean and covariance functions associated with the k^{th} gating function. Similarly to the experts, dependence on the inputs and hyperparam-

eters is dropped from the gating network's GP prior, i.e. $p(\mathbf{h}(\mathbf{X})) = p(\mathbf{h}(\mathbf{X}) \mid \mathbf{X}, \boldsymbol{\phi})$. In contrast to the experts, partitioning the data set is not desirable for the gating network GPs, as each gating function should depend on all of the training observations.

Each mode's mixing probability $\Pr(\alpha_n = k \mid \mathbf{x}_n, \boldsymbol{\phi})$ is then obtained by marginalising **all** of the gating functions. In the general case where $\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n))$ uses the softmax function (Equation (3.13)) this integral is intractable, so it is approximated with Monte Carlo quadrature.

Bernoulli ($K = 2$) Instantiating the model with two experts, $\alpha_n \in \{1, 2\}$, is a special case where only a single gating function is needed. This is because the output of a function $h(\mathbf{x}_n)$ can be mapped through a sigmoid function $\text{sig} : \mathbb{R} \to [0, 1]$ and interpreted as a probability,

$$\Pr(\alpha_n = 1 \mid h_1(\mathbf{x}_n)) = \text{sig}(h_1(\mathbf{x}_n)). \tag{3.15}$$

If this sigmoid function satisfies the point symmetry condition then the following holds, $\Pr(\alpha_n = 2 \mid h_1(\mathbf{x}_n)) = 1 - \Pr(\alpha_n = 1 \mid h_1(\mathbf{x}_n))$. This only requires a single gating function and no normalisation term needs to be calculated. If the sigmoid function in Equation (3.15) is selected to be the Gaussian cumulative distribution function $\Phi(h(\cdot)) = \int_{-\infty}^{h(\cdot)} \mathcal{N}(\tau|0, 1)\mathrm{d}\tau$, then the mixing probability can be calculated in closed-form,

$$\begin{aligned} \Pr(\alpha_n = 1 \mid \mathbf{x}_n, \boldsymbol{\phi}) &= \int \Phi(h(\mathbf{x}_n))\mathcal{N}\big(h(\mathbf{x}_n) \mid \mu_h, \sigma_h^2\big)\mathrm{d}h(\mathbf{x}_n) \\ &= \Phi\left(\frac{\mu_h}{\sqrt{1 + \sigma_h^2}}\right), \end{aligned} \tag{3.16}$$

where μ_h and σ_h^2 represent the mean and variance of the gating GP at $\mathbf{x}_n$ respectively.

3.4 Approximate Inference

> Nature laughs at the difficulties of integration.
>
> *Pierre-Simon Laplace*

Performing Bayesian inference involves finding the posterior over the latent variables,

$$p(\mathbf{h}(\mathbf{X}), \mathbf{f}(\mathbf{X}) \mid \mathbf{X}, \mathbf{y}, \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{\sum_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha} \mid \mathbf{h}(\mathbf{X})) p(\mathbf{h}(\mathbf{X})) \prod_{k=1}^{K} p(\mathbf{y}_k \mid f_k(\mathbf{X}_k)) p(f_k(\mathbf{X}_k))}{p(\mathbf{y} \mid \mathbf{X})} \tag{3.17}$$

where the denominator is the marginal likelihood from Equation (3.11). Exact inference in our model is intractable due to the marginalisation over the set of expert indicator variables. For this reason, we resort to a variational approximation. The rich structure of our model makes it hard to construct an ELBO that can be evaluated in closed form whilst accurately modelling the complex dependencies. Further to this, the marginal likelihood is extremely expensive to evaluate, as there are K^N sets of assignments $\boldsymbol{\alpha}$ that need to be marginalised. For each set of assignments, K GP experts that need to be evaluated, each with complexity $\mathcal{O}(N^3)$. For these reasons, this work derives a variational approximation based on inducing variables, that provides scalability by utilising stochastic gradient-based optimisation.

Stochastic Variational Inference (SVI) (Hoffman et al., 2013) relies upon having a set of local variables factorised across observations and a set of global variables. The marginalisation over the set of expert indicator variables $\boldsymbol{\alpha}$ in Equation (3.7) is prohibitive to SVI. This is because SVI requires a set of local variables factorised over data but the marginalisation considers the sets of assignments for the entire data set. Following the approach by Titsias, 2009, we augment the probability space with a set of M inducing variables for each GP. However, instead of collapsing these inducing variables, we explicitly represent them as variational distributions and use

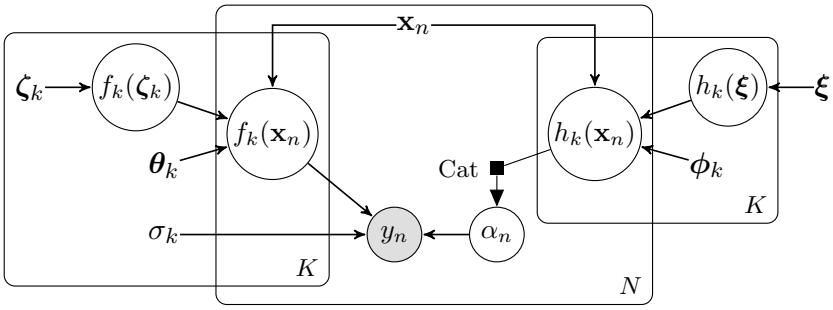


Figure 3.2: *Graphical model of the augmented probability space where the joint distribution over the data is captured by the inducing variables $f_k(\boldsymbol{\zeta}_k)$ and $h_k(\boldsymbol{\xi})$. The observations assigned to expert k are modelled by the inducing points $\{\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k)\}_{k=1}^K$. This model avoids the hard assignment of observations to experts by letting the gating network softly assign them in the ELBO.*

them to lower bound (and then further bound) the marginal likelihood, similar to Hensman et al., 2013, 2015. Figure 3.2 shows the graphical model of the augmented joint probability space.

Augmented experts We sidestep the hard assignment of observations to experts by augmenting each expert with a set of separate independent inducing points $(\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k))$. Each expert's inducing points are assumed to be from its GP prior,

$$p(\mathbf{f}(\boldsymbol{\zeta})) = \prod_{k=1}^{K} p(f_k(\boldsymbol{\zeta}_k)) = \prod_{k=1}^{K} \mathcal{N}(f_k(\boldsymbol{\zeta}_k) \mid \mu_k(\boldsymbol{\zeta}_k), k_k(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_k)), \tag{3.18}$$

where the set of all inducing inputs associated with the experts has been denoted $\boldsymbol{\zeta}$ and the set of all inducing variables as $\mathbf{f}(\boldsymbol{\zeta})$. Note that the dependence on the inducing inputs has been dropped for notational conciseness. Introducing separate inducing points from each expert's GP can loosely be seen as "partitioning" the observations between experts. However, as the assignment of observations to experts is *not known a priori*, the inducing inputs $\boldsymbol{\zeta}_k$ and variables $f_k(\boldsymbol{\zeta}_k)$, must be inferred from observations.

Augmented gating network Following a similar approach for the gating network, each gating function is augmented with a set of M inducing points from its corresponding GP prior,

$$p(\mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} p(h_k(\boldsymbol{\xi})) = \prod_{k=1}^{K} \mathcal{N}\Big(h_k(\boldsymbol{\xi}) \mid \hat{\mu}_k(\boldsymbol{\xi}), \hat{k}_k(\boldsymbol{\xi}, \boldsymbol{\xi})\Big), \tag{3.19}$$

where $h_k(\boldsymbol{\xi})$ are the inducing variables associated with the k^{th} gating function. Again, the dependence on the inducing inputs has been dropped for notational conciseness. The set of all inducing variables associated with the gating network has been denoted $\mathbf{h}(\boldsymbol{\xi})$. Partitioning of the data set is not desirable for the gating function GPs as each gating function should depend on all of the training observations. For this reason, the gating functions share the same inducing inputs $\boldsymbol{\xi}$.

Marginal likelihood These inducing points are used to approximate the marginal likelihood with a factorisation over observations that is favourable for constructing a GP-based gating network. Our approximate marginal likelihood is given by,

$$p(\mathbf{y} \mid \mathbf{X}) \approx \mathbb{E}_{p(\mathbf{h}(\boldsymbol{\xi}))p(\mathbf{f}(\boldsymbol{\zeta}))}\left[\prod_{n=1}^{N}\sum_{k=1}^{K} \Pr(\alpha_n = k \mid \mathbf{h}(\boldsymbol{\xi}))p(y_n \mid f_k(\boldsymbol{\zeta}_k))\right], \tag{3.20}$$

where the conditional distributions $p(y_n \mid f_k(\boldsymbol{\zeta}_k))$ and $\Pr(\alpha_n = k \mid \mathbf{h}(\boldsymbol{\xi}))$ follow from standard sparse GP methodologies,

$$p(y_n \mid f_k(\boldsymbol{\zeta}_k)) = \mathbb{E}_{p(f_k(\mathbf{x}_n)\mid f_k(\boldsymbol{\zeta}_k))}[p(y_n \mid f_k(\mathbf{x}_n))] \tag{3.21}$$

$$\Pr(\alpha_n = k \mid \mathbf{h}(\boldsymbol{\xi})) = \mathbb{E}_{p(\mathbf{h}(\mathbf{x}_n)\mid \mathbf{h}(\boldsymbol{\xi}))}[\Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n))] \tag{3.22}$$

$$p(f_k(\mathbf{x}_n) \mid f_k(\boldsymbol{\zeta}_k)) = \mathcal{N}\big(f_k(\mathbf{x}_n) \mid \mathbf{k}_{knM}\mathbf{K}_{kMM}^{-1} f_k(\boldsymbol{\zeta}_k), k_{knn} - \mathbf{k}_{knM}\mathbf{K}_{kMM}^{-1}\mathbf{k}_{kMn}\big), \tag{3.23}$$

$$p(\mathbf{h}(\mathbf{x}_n) \mid \mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} \mathcal{N}\Big(h_k(\mathbf{x}_n) \mid \hat{\mathbf{k}}_{knM}\hat{\mathbf{K}}_{kMM}^{-1} h_k(\boldsymbol{\xi}), \hat{k}_{knn} - \hat{\mathbf{k}}_{knM}\hat{\mathbf{K}}_{kMM}^{-1}\hat{\mathbf{k}}_{kMn}\Big), \tag{3.24}$$

where $\mathbf{K}_{kMM} = k_k(\boldsymbol{\zeta}_k, \boldsymbol{\zeta}_k)$ represents the k^{th} expert's kernel evaluated between its inducing inputs, $k_{knn} = k_k(\mathbf{x}_n, \mathbf{x}_n)$ represents it evaluated between the n^{th} training input and $\mathbf{k}_{knM} = k_k(\mathbf{x}_n, \boldsymbol{\zeta}_k)$ between the n^{th} training input and its inducing inputs.

Similarly for the gating network. Figure 3.2 shows the graphical model of this augmented joint probability model.

Our work follows from sparse GP methodologies that assume, given the inducing variables, the latent function values factorise over observations. Our approximation assumes that given the inducing points, the marginalisation over every possible assignment of data points to experts can be factorised over data. In a similar spirit to the Fully Independent Training Conditional (FITC) approximation (Naish-guzman and Holden, 2008; Quiñonero-Candela and Rasmussen, 2005), this can be viewed as a likelihood approximation,

$$p(\mathbf{y} \mid \mathbf{f}(\boldsymbol{\zeta})) \approx \prod_{n=1}^{N} p(y_n \mid \mathbf{f}(\boldsymbol{\zeta})) = \prod_{n=1}^{N} \sum_{k=1}^{K} \Pr(\alpha_n = k \mid \mathbf{x}_n, \boldsymbol{\phi}) \prod_{k=1}^{K} p(y_n \mid f_k(\boldsymbol{\zeta}_k)). \tag{3.25}$$

Importantly, the factorisation over observations has been moved outside of the marginalisation over the expert indicator variable, i.e. the expert indicator variable can be marginalised for each data point separately. This approximation assumes that the inducing variables, $\{f_k(\boldsymbol{\zeta}_k)\}_{k=1}^{K}$, are a sufficient statistic for their associated latent function values, $\{f_k(\mathbf{X}_k)\}_{k=1}^{K}$ and the set of assignments $\boldsymbol{\alpha}$. This approximation becomes exact in the limit $KM = N$, if each expert's inducing points represent the true data partition $\{\boldsymbol{\zeta}_k, f_k(\boldsymbol{\zeta}_k)\}_{k=1}^{K} = \{\mathbf{X}_k, f_k(\mathbf{X}_k)\}_{k=1}^{K}$. It is also worth noting that Equation (3.20) captures a rich approximation of each expert's covariance but as $KM \ll N$ the computational complexity is much lower. This approximation efficiently couples the gating network and the experts by marginalising the expert indicator variable for each data point separately.

Our approximate marginal likelihood captures the joint distribution over the data and assignments through the inducing variables $\mathbf{f}(\boldsymbol{\zeta})$. As such, information regarding the assignment of observations to experts must pass through the bottleneck of the inducing variables. This approximation induces a local factorisation over observations and a set of global variables – the necessary conditions for SVI. This

approach can loosely be viewed as parameterising the full nonparametric model in Equation (3.6) to obtain a desirable factorisation for 1) constructing a GP-based gating network and 2) deriving an ELBO that can be optimised with stochastic gradient methods, whilst still capturing the complex dependencies between the gating network and experts.

3.4.1 Evidence Lower Bounds

Instead of collapsing the inducing variables as seen in Titsias, 2009, they can be explicitly represented as variational distributions, $(q(\mathbf{f}(\boldsymbol{\zeta})), q(\mathbf{h}(\boldsymbol{\xi})))$ and used to obtain a variational lower bound, aka Evidence Lower Bound (ELBO). This section derives three ELBOs. The first bound is the tightest but requires approximating M dimensional integrals for each expert and gating function. Two further lower bounds which replace some (or all) of the M dimensional integrals with one-dimensional integrals are derived. These further bounds offer improved computational properties at the cost of loosening the bound. All three of these bounds are evaluated in Section 3.5.2.

Tight lower bound Following a similar approach to Hensman et al., 2013, 2015, a lower bound on Equation (3.20) can be obtained,

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) \geq \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{h}(\boldsymbol{\xi}))q(\mathbf{f}(\boldsymbol{\zeta}))}\left[\log\left(\sum_{k=1}^{K} \Pr(\alpha_n = k \mid \mathbf{h}(\boldsymbol{\xi}))p(y_n \mid f_k(\boldsymbol{\zeta}_k))\right)\right] \\ - \sum_{k=1}^{K} \mathrm{KL}(q(f_k(\boldsymbol{\zeta}_k)) \mid\mid p(f_k(\boldsymbol{\zeta}_k))) \\ - \sum_{k=1}^{K} \mathrm{KL}(q(h_k(\boldsymbol{\xi})) \mid\mid p(h_k(\boldsymbol{\xi}))) := \mathcal{L}_{\text{tight}},\end{aligned} \tag{3.26}$$

where we parameterise the variational posteriors to be independent Gaussians,

$$q(\mathbf{f}(\boldsymbol{\zeta})) = \prod_{k=1}^{K} q(f_k(\boldsymbol{\zeta}_k)) = \prod_{k=1}^{K} \mathcal{N}(f_k(\boldsymbol{\zeta}_k) \mid \mathbf{m}_k, \mathbf{S}_k) \tag{3.27}$$

$$q(\mathbf{h}(\boldsymbol{\xi})) = \prod_{k=1}^{K} q(h_k(\boldsymbol{\xi})) = \prod_{k=1}^{K} \mathcal{N}\Big(h_k(\boldsymbol{\xi}) \mid \hat{\mathbf{m}}_k, \hat{\mathbf{S}}_k\Big). \tag{3.28}$$

The bound in Equation (3.26) meets the necessary conditions to perform stochastic gradient methods on $q(\mathbf{f}(\boldsymbol{\zeta}))$ and $q(\mathbf{h}(\boldsymbol{\xi}))$, as the expected log likelihood (first term) is written as a sum over input-output pairs. However, this expectation cannot be calculated in closed form and must be approximated. The joint distributions over the inducing variables for each expert GP $q(f_k(\boldsymbol{\zeta}_k))$ and each gating function GP $q(h_k(\boldsymbol{\xi}))$, are M dimensional multivariate normal distributions. Therefore, each expectation requires an M dimensional integral to be approximated.

Further lower bound Following Hensman et al., 2015, these issues can be overcome by further bounding $\mathcal{L}_{\text{tight}}$ from Equation (3.26). This removes the M dimensional integrals associated with each of the gating functions. Jensen's inequality can be applied to the conditional probability $p(h_k(\mathbf{x}_n) \mid h_k(\boldsymbol{\xi}))$, obtaining the further bound,

$$\begin{aligned}\mathcal{L}_{\text{tight}} \geq \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{h}(\mathbf{x}_n))q(\mathbf{f}(\boldsymbol{\zeta}))}\left[\log \sum_{k=1}^{K} \Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n)) p(y_n \mid f_k(\boldsymbol{\zeta}_k))\right] \\ - \sum_{k=1}^{K} \mathrm{KL}(q(f_k(\boldsymbol{\zeta}_k)) \,||\, p(f_k(\boldsymbol{\zeta}_k))) \\ - \sum_{k=1}^{K} \mathrm{KL}(q(h_k(\boldsymbol{\xi})) \,||\, p(h_k(\boldsymbol{\xi}))) := \mathcal{L}_{\text{further}},\end{aligned} \tag{3.29}$$

where $q(\mathbf{h}(\mathbf{x}_n))$ represents the variational posterior given by,

$$q(\mathbf{h}(\mathbf{x}_n)) = \prod_{k=1}^{K} \mathbb{E}_{q(h_k(\boldsymbol{\xi}))}[p(h_k(\mathbf{x}_n) \mid h_k(\boldsymbol{\xi}))]. \tag{3.30}$$

Moving the marginalisation over the latent gating functions $\mathbf{h}(\mathbf{x}_n)$ outside of the marginalisation over the expert indicator variable is possible because the mixing probabilities are dependent on **all** of the gating functions and not just their associated gating function. In contrast, moving the marginalisation over each expert's latent function $f_k(\mathbf{x}_n)$ outside of the marginalisation over the expert indicator variable corresponds to changing the underlying model, in particular, the likelihood approximation in Equation (3.25).

Further^2 lower bound Nevertheless, we proceed and further bound the experts for comparison. Jensen's inequality is applied to the conditional probability $p(f_k(\mathbf{x}_n) \mid f_k(\boldsymbol{\zeta}_k))$, obtaining the further^2 bound,

$$\begin{aligned}\mathcal{L}_{\text{further}} \geq \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{h}(\mathbf{x}_n))q(\mathbf{f}(\mathbf{x}_n))}\left[\log \sum_{k=1}^{K} \Pr(\alpha_n = k \mid \mathbf{h}(\mathbf{x}_n)) p(y_n \mid f_k(\mathbf{x}_n))\right] \\ - \sum_{k=1}^{K} \mathrm{KL}(q(f_k(\boldsymbol{\zeta}_k)) \,||\, p(f_k(\boldsymbol{\zeta}_k))) \\ - \sum_{k=1}^{K} \mathrm{KL}(q(h_k(\boldsymbol{\xi})) \,||\, p(h_k(\boldsymbol{\xi}))) := \mathcal{L}_{\text{further}^2},\end{aligned} \tag{3.31}$$

where $q(\mathbf{f}(\mathbf{x}_n))$ represents the variational posterior given by,

$$q(\mathbf{f}(\mathbf{x}_n)) = \prod_{k=1}^{K} \mathbb{E}_{q(f_k(\boldsymbol{\zeta}_k))}[p(f_k(\mathbf{x}_n) \mid f_k(\boldsymbol{\zeta}_k))]. \tag{3.32}$$

Intuitively, this bound can be seen as modifying the likelihood approximation in Equation (3.25). Instead of mixing the GPs associated with each expert, this approximation simply mixes their associated noise models.

As each GP's inducing variables are normally distributed, the functional form of the variational posteriors are given by,

$$q(\mathbf{f}(\mathbf{x}_n)) = \prod_{k=1}^{K} \mathcal{N}\big(f_k(\mathbf{x}_n) \mid \mathbf{A}_k \mathbf{m}_k, k_{knn} + \mathbf{A}_k(\mathbf{S}_k - \mathbf{K}_{kMM})\mathbf{A}_k^T\big) \tag{3.33}$$

$$q(\mathbf{h}(\mathbf{x}_n)) = \prod_{k=1}^{K} \mathcal{N}\Big(h_k(\mathbf{x}_n) \mid \hat{\mathbf{A}}_k \hat{\mathbf{m}}_k, \hat{k}_{knn} + \hat{\mathbf{A}}_k(\hat{\mathbf{S}}_k - \hat{\mathbf{K}}_{kMM})\hat{\mathbf{A}}_k^T\Big), \tag{3.34}$$

where $\mathbf{A}_k = \mathbf{k}_{knM}\mathbf{K}_{kMM}^{-1}$ and $\hat{\mathbf{A}}_k = \hat{\mathbf{k}}_{knM}\hat{\mathbf{K}}_{kMM}^{-1}$. Importantly, these variational posteriors marginalise the inducing variables in closed form, with Gaussian convolutions. $\mathcal{L}_{\text{further}}$ removes K of the undesirable approximate M dimensional integrals from Equation (3.26) and $\mathcal{L}_{\text{further}^2}$ removes K^2. The variational expectation in Equation (3.29) still requires approximation, however, the integrals are now only one-dimensional. These integrals are approximated with Gibbs sampling and in practice only single samples are used because the added stochasticity helps the optimisation.

The tight lower bound $\mathcal{L}_{\text{tight}}$ is the most accurate lower bound, but it is also the most computationally expensive. Both the further lower bound $\mathcal{L}_{\text{further}}$ and the further^2 lower bound $\mathcal{L}_{\text{further}^2}$ have lower computational complexity at the cost of being looser bounds. The performance of these bounds is evaluated in Section 3.5.2.

3.4.2 Optimisation

The bounds in Equations (3.26), (3.29) and (3.31) meet the necessary conditions to perform stochastic gradient methods on $q(\mathbf{f}(\boldsymbol{\zeta}))$ and $q(\mathbf{h}(\boldsymbol{\xi}))$. Firstly, they contain a sum of N terms corresponding to input-output pairs, enabling optimisation with mini-batches. Secondly, the expectations over the log-likelihood are calculated using Monte Carlo samples.

Stochastic optimisation At each iteration j, a random subset of N_b data points are sampled from the data set $\mathcal{D}$, to get a minibatch $\mathcal{D}_j = \{\mathbf{x}_i, y_i\}_{i=1}^{N_b}$. The further lower bound is then approximated by,

$$\hat{\mathcal{L}}_{\text{further}} = \frac{N}{N_b} \sum_{\mathbf{x}_i, y_i \in \mathcal{D}_j} \left(\frac{1}{S} \sum_{s=1}^{S} \frac{1}{\hat{S}} \sum_{\hat{s}=1}^{\hat{S}} \log \sum_{k=1}^{K} \Pr\Big(\alpha_i = k \mid \mathbf{h}(\mathbf{x}_i)^{(\hat{s})}\Big) p\Big(y_i \mid f_k(\boldsymbol{\zeta}_k)^{(s)}\Big) \right) - \sum_{k=1}^{K} \mathrm{KL}(q(h_k(\boldsymbol{\xi})) \,||\, p(h_k(\boldsymbol{\xi}))) - \sum_{k=1}^{K} \mathrm{KL}(q(f_k(\boldsymbol{\zeta}_k)) \,||\, p(f_k(\boldsymbol{\zeta}_k))), \tag{3.35}$$

where $f_k(\boldsymbol{\zeta}_k)^{(s)} \sim q(f_k(\boldsymbol{\zeta}_k))$ and $\mathbf{h}(\mathbf{x}_i)^{(\hat{s})} \sim q(\mathbf{h}(\mathbf{x}_i))$ denote samples from the variational posteriors. $\hat{S}$ samples are drawn from the gating network's variational posterior and S samples are drawn from the experts' variational posterior. The variational distributions over the inducing variables are represented using the mean vector $\mathbf{m}_k$ and the lower triangular $\mathbf{L}_k$ of the covariance matrix $\mathbf{S}_k = \mathbf{L}_k \mathbf{L}_k^T$. A downside to this formulation is that $(K^2)(M-1)M/2 + K^2 M$ extra parameters need to be optimised. In the two expert case, this reduces to $(K+1)(M-1)M/2 + (K+1)M$ extra parameters. Optimising the inducing inputs ($\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$) introduces a further $K^2 MD$ optimisation parameters. The inducing inputs $\boldsymbol{\xi}, \{\boldsymbol{\zeta}_k\}_{k=1}^K$, kernel hyperparameters and noise variances, are treated as variational hyperparameters and optimised alongside the variational parameters, using stochastic gradient descent e.g. Adam (Kingma and Ba, 2017).

Computational complexity Assuming that each expert has the same number of inducing points M, the cost of computing the KL divergences and their derivatives is $\mathcal{O}\big(KM^3\big)$. The cost of computing the expected likelihood term is dependent on the batch size N_b. For each data point in the minibatch, each of the K gating function variational posteriors has complexity $\mathcal{O}\big(M^2\big)$ to evaluate. For each data point, only a single sample is drawn from each of these distributions. Sampling each expert's inducing variable distribution $q(f_k(\boldsymbol{\zeta}_k))$ has complexity $\mathcal{O}(M^2)$ because the covariance is represented as the lower triangular (via the Cholesky decomposition).

In addition to this sampling, calculating each expert's conditional $p(y_n \mid f_k(\boldsymbol{\zeta}_k))$ given these samples has complexity $\mathcal{O}(M^2)$.

3.4.3 Predictions

For a given set of test inputs $\mathbf{X}^* \in \mathbb{R}^{N^* \times D}$, this model makes probabilistic predictions following a mixture of K Gaussians. Making predictions with this model involves calculating a density over the output for each expert and combining them using the probabilities obtained from the gating network, i.e. marginalising the expert indicator variable. This is given by,

$$p(\mathbf{y}^* \mid \mathbf{y}) = \prod_{n=1}^{N^*} \sum_{k=1}^{K} \underbrace{\Pr(\alpha_n^* = k \mid \mathbf{y})}_{\text{gating network posterior}} \underbrace{p(y_n^* \mid \alpha_n^* = k, \mathbf{y})}_{\text{expert } K \text{ posterior}} \tag{3.36}$$

$$\approx \prod_{n=1}^{N^*} \sum_{k=1}^{K} q(\alpha_n^* = k) q(y_n^* \mid \alpha_n^* = k), \tag{3.37}$$

where dependence on the test inputs $\mathbf{X}^*$ and the training inputs $\mathbf{X}$ have been dropped for notational conciseness.

Experts The experts make predictions at new test locations by integrating over their latent function posteriors,

$$\begin{aligned} p(y_n^* \mid \alpha_n^* = k, \mathbf{y}) &= \int \underbrace{p(y_n^* \mid f_k(\mathbf{x}_n^*))}_{\text{likelihood}} \underbrace{p(f_k(\mathbf{x}_n^*) \mid \mathbf{y})}_{\text{posterior}} \mathrm{d}f_k(\mathbf{x}_n^*) \\ &\approx \int \underbrace{p(y_n^* \mid f_k(\mathbf{x}_n^*))}_{\text{likelihood}} \underbrace{q(f_k(\mathbf{x}_n^*))}_{\text{approx posterior}} \mathrm{d}f_k(\mathbf{x}_n^*) \coloneqq q(y_n^* \mid \alpha_n^* = k). \end{aligned} \tag{3.38}$$

However, the experts' true posteriors $p(f_k(\mathbf{x}_n^*) \mid \mathbf{y})$ are not known and have been approximated. Each expert's approximate posterior is given by $q(f_k(\mathbf{X}_k), f_k(\boldsymbol{\zeta}_k)) = p(f_k(\mathbf{X}_k) \mid f_k(\boldsymbol{\zeta}_k)) q(f_k(\boldsymbol{\zeta}_k))$. To make a prediction at a set of test locations $\mathbf{X}^*$, we substitute our approximate posterior into the standard probabilistic rule,

$$\begin{aligned}
\underbrace{p(f_k(\mathbf{x}_n^*) \mid \mathbf{y})}_{\text{posterior}} &= \int p(f_k(\mathbf{x}_n^*) \mid f_k(\mathbf{X}_k), f_k(\boldsymbol{\zeta}_k)) p(f_k(\mathbf{X}_k), f_k(\boldsymbol{\zeta}_k) \mid \mathbf{y}) \mathrm{d}f_k(\mathbf{X}_k) \mathrm{d}f_k(\boldsymbol{\zeta}_k) \\
&\approx \int p(f_k(\mathbf{x}_n^*) \mid f_k(\mathbf{X}_k), f_k(\boldsymbol{\zeta}_k)) p(f_k(\mathbf{X}_k) \mid f_k(\boldsymbol{\zeta}_k)) q(f_k(\boldsymbol{\zeta}_k)) \mathrm{d}f_k(\mathbf{X}_k) \mathrm{d}f_k(\boldsymbol{\zeta}_k) \\
&= \int p(f_k(\mathbf{x}_n^*) \mid f_k(\boldsymbol{\zeta}_k)) q(f_k(\boldsymbol{\zeta}_k)) \mathrm{d}f_k(\boldsymbol{\zeta}_k) \\
&= \mathcal{N}\big(f_k(\mathbf{x}_n^*) \mid \mathbf{A}_k \mathbf{m}_k, k_{k**} + \mathbf{A}_k(\mathbf{S}_k - \mathbf{K}_{kMM})\mathbf{A}_k^T\big) \coloneqq \underbrace{q(f_k(\mathbf{x}_n^*))}_{\text{approx posterior}} ,
\end{aligned} \tag{3.39}$$

where $\mathbf{A}_k = \mathbf{k}_{k*M}\mathbf{K}_{kMM}^{-1}$. This integral has complexity $\mathcal{O}(M^2)$.

Gating network The mixing probabilities associated with the gating network are obtained by integrating the gating network's posterior through the gating likelihood,

$$\begin{aligned}
\Pr(\alpha_n^* = k \mid \mathbf{y}) &= \int \underbrace{\Pr(\alpha_n^* = k \mid \mathbf{h}(\mathbf{x}_n^*))}_{\text{likelihood}} \underbrace{p(\mathbf{h}(\mathbf{x}_n^*) \mid \mathbf{y})}_{\text{posterior}} \mathrm{d}\mathbf{h}(\mathbf{x}_n^*) \\
&\approx \int \underbrace{\Pr(\alpha_n^* = k \mid \mathbf{h}(\mathbf{x}_n^*))}_{\text{likelihood}} \underbrace{q(\mathbf{h}(\mathbf{x}_n^*)}_{\text{approx posterior}} \mathrm{d}\mathbf{h}(\mathbf{x}_n^*) \coloneqq q(\alpha_n^* = k).
\end{aligned} \tag{3.40}$$

Again, the gating network's true posterior $p(\mathbf{h}(\mathbf{x}_n^*) \mid \mathbf{y})$ has been approximated,

$$\begin{aligned}
\underbrace{p(\mathbf{h}(\mathbf{x}_n^*) \mid \mathbf{y})}_{\text{posterior}} &\approx \int p(\mathbf{h}(\mathbf{x}_n^*) \mid h_k(\boldsymbol{\xi})) q(h_k(\boldsymbol{\xi})) \mathrm{d}h_k(\boldsymbol{\xi}) \\
&= \prod_{k=1}^{K} \mathcal{N}\Big(\mathbf{h}(\mathbf{x}_n^*) \mid \hat{\mathbf{A}}_k \hat{\mathbf{m}}_k, \hat{k}_{k**} + \hat{\mathbf{A}}_k(\hat{\mathbf{S}}_k - \hat{\mathbf{K}}_{kMM})\hat{\mathbf{A}}_k^T\Big) \coloneqq \underbrace{q(\mathbf{h}(\mathbf{x}_n^*)}_{\text{approx posterior}} ,
\end{aligned} \tag{3.41}$$

where $\hat{\mathbf{A}}_k = \hat{\mathbf{k}}_{k*M}\hat{\mathbf{K}}_{kMM}^{-1}$. In the general case, when $K > 2$, the gating likelihood is the softmax,

$$\underbrace{\Pr(\alpha_n^* = k \mid \mathbf{h}(\mathbf{x}_n^*))}_{\text{likelihood}} = \text{softmax}_k(\mathbf{h}(\mathbf{x}_n^*)), \tag{3.42}$$

so Equation (3.40) is approximated with Monte Carlo quadrature. In the two expert case there is only a single gating function, h_1, as,

$$\Pr(\alpha_n^* = 2 \mid h_1(\mathbf{x}_n^*)) = 1 - \Pr(\alpha_n^* = 1 \mid h_1(\mathbf{x}_n^*)). \tag{3.43}$$

In this case, the gating likelihood is the Gaussian cdf,

$$\underbrace{\Pr(\alpha_n^* = 1 \mid h_1(\mathbf{x}_n^*))}_{\text{likelihood}} = \Phi(h_1(\mathbf{x}_n^*)), \tag{3.44}$$

so Equation (3.40) can be calculated in closed-form with,

$$\begin{aligned}\Pr(\alpha_n^* = 1 \mid \mathbf{y}) &= \int q(h_1(\mathbf{x}_n^*))\Phi(h_1(\mathbf{x}_n^*))\mathrm{d}h_1(\mathbf{x}_n^*) \\ &= \Phi\left(\frac{\mu_{h*}}{\sqrt{1+\sigma_{h_*}^2}}\right).\end{aligned} \tag{3.45}$$

where μ_{h_*} and $\sigma_{h_*}^2$ are the mean and variance of the variational posterior $q(h_1(\mathbf{x}_n^*))$ at $\mathbf{x}_n^*$.

3.5 Evaluation of Model and Approximate Inference

As a Mixture of Experts (MoE) method, our model aims to improve on standard GP regression with the ability to model non-stationary functions and multimodal distributions over the output variable. With this in mind, the model and approximate inference scheme are evaluated on two data sets. Following other MoGPE work, they are first tested on the motorcycle data set (Silverman, 1985). Although this data

set does not represent state transitions from a dynamical system, it does contain non-stationary points and heterogeneous noise, making it interesting to study from the MoGPE perspective. Secondly, they are tested on the illustrative example from Section 1.1. That is, a data set collected onboard a DJI Tello quadcopter flying in an environment subject to two dynamics modes.

3.5.1 Experiments

All data sets were split into test and training sets with 70% for training and 30% for testing. In order to evaluate and compare the full predictive posteriors the Negative Log Predictive Probability (NLPP) is computed on the test set. The models are also compared using the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE). Given a test data set $(\mathbf{X}^*, \mathbf{y}^*) = \{(\mathbf{x}_n^*, y_n^*)\}_{n=1}^{N^*}$, they are calculated as follows,

$$\text{RMSE} = \sqrt{\frac{1}{N^*}\sum_{n=1}^{N^*}(\hat{y}_n^* - y_n^*)^2} \tag{3.46}$$

$$\text{MAE} = \frac{1}{N^*}\sum_{n=1}^{N^*}|\hat{y}_n^* - y_n^*| \tag{3.47}$$

$$\text{NLPP} = \frac{1}{N^*}\sum_{n=1}^{N^*} -\log(y_n^* \mid \mathbf{x}_n^*, \mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\phi}) \tag{3.48}$$

where $\hat{y}_n^*$ is the model's prediction at $\mathbf{x}_n^*$. Note that all figures in this section show models that were trained on the full data set, i.e. no test/train split.

3.5.2 Evaluation on Motorcycle Data Set

The Motorcycle data set (discussed in Silverman, 1985) contains 133 data points ($\mathbf{X} \in \mathbb{R}^{133\times 1}$ and $\mathbf{y} \in \mathbb{R}^{133\times 1}$) and input dependent noise. The data set represents motorcycle impact data – time (ms) vs acceleration (g). The data set is represented by the black crosses in Figure 3.3.

To test the performance of MoSVGPE, the model is instantiated with $K = 2$ and $K = 3$ experts. All experiments on the Motorcycle data set use $M = 32$ inducing points for all GPs and are trained for $25,000$ iterations with Adam (Kingma and Ba, 2017), with a learning rate of 0.01 and a batch size of $N_b = 16$. The results are compared against a GP and a Sparse Variational Gaussian Process (SVGP), which use Squared Exponential kernels with Automatic Relevance Determination (ARD) and a Gaussian likelihood.

Table 3.1 summarises the results for the three ELBOs ($\mathcal{L}_{\text{tight}}$, $\mathcal{L}_{\text{further}}$, $\mathcal{L}_{\text{further}^2}$) and compares them to a standard GP regression model and a SVGP method instantiated with $M = 16$ and $M = 32$ inducing points. Both methods use Gaussian likelihoods and optimise their hyperparameters, noise variances (and inducing inputs in the SVGP case) using their well-known objectives – the marginal likelihood and ELBO. The NLPP indicates the probability of the data given the parameters which are

Table 3.1: *Results on the Motorcycle data set (Silverman, 1985) with different instantiations of our model (MoSVGPE). Comparison of the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Negative Log Predictive Probability (NLPP) on the test data set. Results for a GP and a SVGP with $M = 16$ and $M = 32$ inducing points are shown for comparison. All models were instantiated with Squared Exponential kernels and were trained for $25,000$ iterations. The GP's hyperparameters were optimised using SciPy's (Virtanen et al., 2020) L-BFGS-B optimiser. The SVGP and MoSVGPE models were trained with Adam (Kingma and Ba, 2017) using a learning rate of 0.01 and a minibatch size of $N_b = 16$. The MoSVGPE experiments used $M = 32$ inducing points for each expert GP and each gating function GP.*

	RMSE	NLPP	MAE
GP	**0.4357**	0.9886	0.3242
SVGP ($M = 16$)	0.4427	0.9762	0.3257
SVGP ($M = 32$)	0.4437	0.9832	0.3271
MoSVGPE ($k = 2, \mathcal{L}_{\text{tight}}$)	0.4442	0.4863	0.3260
MoSVGPE ($k = 2, \mathcal{L}_{\text{further}}$)	0.4590	0.5073	0.3355
MoSVGPE ($k = 2, \mathcal{L}_{\text{further}^2}$)	0.4472	0.5271	**0.3218**
MoSVGPE ($k = 3, \mathcal{L}_{\text{tight}}$)	0.4569	**0.2634**	0.3301
MoSVGPE ($k = 3, \mathcal{L}_{\text{further}}$)	0.4866	0.2695	0.3449
MoSVGPE ($k = 3, \mathcal{L}_{\text{further}^2}$)	0.4575	0.5467	0.3270

not marginalised, e.g. hyperparameters and inducing inputs. Following Bayesian model selection, it is known that lower values indicate higher performing models,

i.e. predictive posteriors that more accurately match the distribution of the data. The predictive posterior is most accurate when MoSVGPE is instantiated with three experts $K = 3$ and trained using the tight lower bound $\mathcal{L}_{\text{tight}}$. In both the two and three expert experiments, the tight lower bound $\mathcal{L}_{\text{tight}}$ achieved better NLPP than both of the further/further^2 lower bounds, $\mathcal{L}_{\text{further}}$ and $\mathcal{L}_{\text{further}^2}$. This is expected as it is a tighter bound. As both of the further/further^2 lower bounds offer improved computational properties, it is interesting to compare their performance. The NLPP scores for the further lower bound $\mathcal{L}_{\text{further}}$ are almost equal to the tight lower bound $\mathcal{L}_{\text{tight}}$. In contrast, the NLPP score in the three expert experiment for the further^2 lower bound $\mathcal{L}_{\text{further}^2}$ is significantly worse. This indicates that valuable information is lost in this bound. This was expected as this bound corresponds to a further likelihood approximation, which mixes the experts' noise models as opposed to their full SVGP models.

Regarding the accuracy of the predictive means, the standard GP regression model achieved the best RMSE, followed by the SVGP models and then the MoSVGPE models. It is worth noting that all of the RMSE and MAE scores are very similar. Although adding more experts to the MoSVGPE model appears to learn more accurate predictive posteriors, the predictive means appear to deteriorate ever so slightly (indicated by higher RMSE/MAE values). This is most likely due to bias at the boundaries between the experts, resulting from the mixing behaviour arising from our GP-based gating network. If the gating functions do not have low lengthscales then they will not be able to immediately switch from one expert to another. It is worth noting that this drop in performance is negligible.

Two Experts

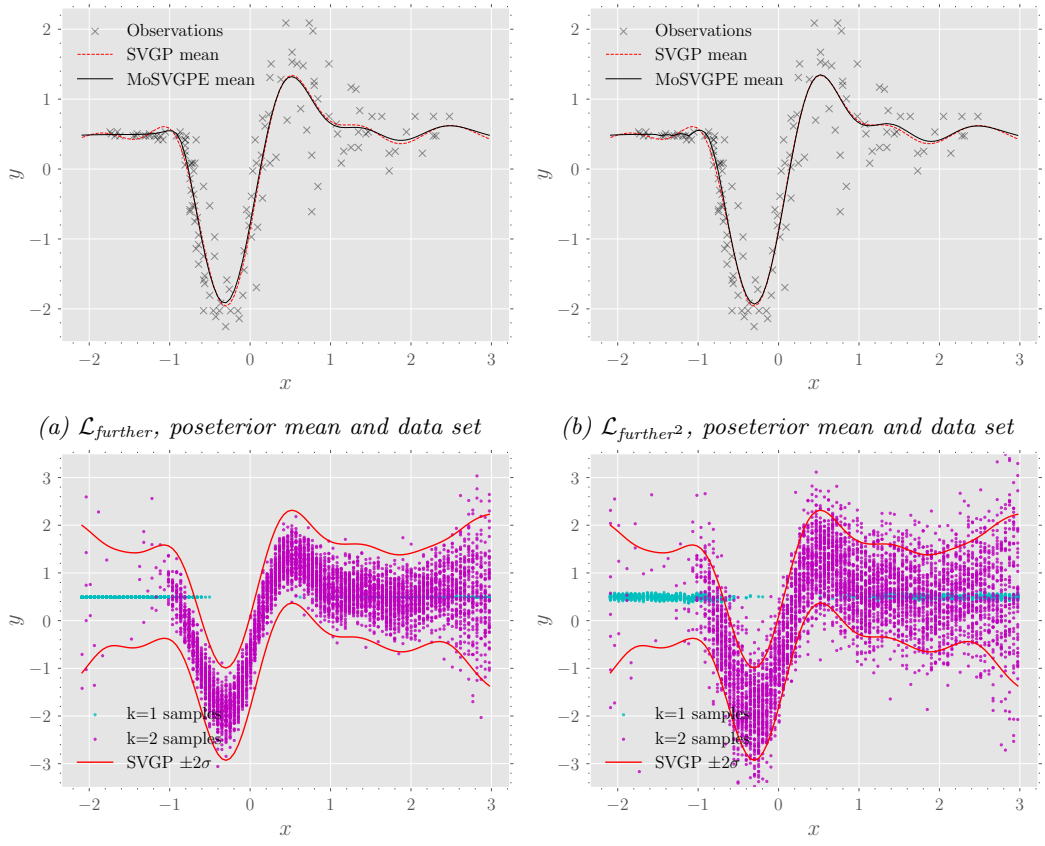
The two further lower bounds ($\mathcal{L}_{\text{further}}$ and $\mathcal{L}_{\text{further}^2}$), derived in Section 3.4, are compared by training each instantiation of the model using the same model and training parameters. Table 3.2 contains the initial values for all of the trainable parameters in the model. They are compared by instantiating the model with two

Table 3.2: *Initial parameter settings before training on motorcycle data set with two experts.*

	Description	Symbol	Value
Optimiser	Num data	N	133
	Batch size	N_b	16
	Epochs	N/A	25000
	Num gating samples	$\hat{S}$	1
	Num expert samples	S	1
	Learning rate	N/A	0.01
Expert 1	Kernel variance	σ_f	0.1
	Kernel lengthscales	l	10
	Likelihood variance	σ_n	0.032
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\zeta}_1$	$\boldsymbol{\zeta}_1 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_1 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_1$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_1$	ones plus Gaussian noise
Expert 2	Kernel variance	σ_f	20
	Kernel lengthscales	l	0.5
	Likelihood variance	σ_n	0.9
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\zeta}_2$	$\boldsymbol{\zeta}_2 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_2 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_2$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_2$	ones plus Gaussian noise
Gating function	Kernel variance	σ_f	3
	Kernel lengthscales	l	0.5
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\xi}$	$\boldsymbol{\xi} \subseteq \mathbf{X}$ with $\#\boldsymbol{\xi} = M$
	Inducing variable mean	$\hat{\mathbf{m}}_k$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_k$	10× ones plus Gaussian noise

experts $K = 2$ and comparing their performance. The results are shown in Figures 3.3 and 3.4, where Figure 3.3 visualises the predictive posteriors and Figure 3.4 visualises the posteriors over the latent variables. The left column shows results for $\mathcal{L}_{\text{further}}$ and the right column shows results for $\mathcal{L}_{\text{further}^2}$. This layout is used in Figures 3.3 to 3.6.

Figures 3.3a and 3.3b compare the posterior means (black solid line) to the SVGP's posterior mean (red dashed line) and Figures 3.3c and 3.3d compare the posterior densities to the SVGP. The red lines show plus or minus two standard deviations of the SVGP's posterior variance. As the MoSVGPE posterior is a Gaussian mixture,



(a) $\mathcal{L}_{further}$*, poseterior mean and data set*

(b) $\mathcal{L}_{further^2}$*, poseterior mean and data set*

(c) $\mathcal{L}_{further}$*, posterior samples*

(d) $\mathcal{L}_{further^2}$*, posterior samples*

Figure 3.3: *Visualisation of the model instantiated with* $K = 2$ *experts, after training on the motorcycle data set (Silverman, 1985) with* $\mathcal{L}_{further}$ *(left column) and with* $\mathcal{L}_{further^2}$ *(right column). (a-b) show the data set (black crosses) and the posterior means associated with the MoSVGPE (black solid line) and the SVGP (red dashed line) for comparison. (c-d) show samples from the MoSVGPE posterior where colour indicates the underlying expert (cyan* $K = 1$*, magenta* $K = 2$*) and the red lines show the* ± 2 *standard deviation error (95% confidence interval) of the SVGP posterior. All experiments were trained with* $25,000$ *iterations of Adam (Kingma and Ba, 2017) using a learning rate of* 0.01 *and a minibatch size of* $N_b = 16$*. All GPs use Squared Exponential kernels and* $M = 32$ *inducing points.*

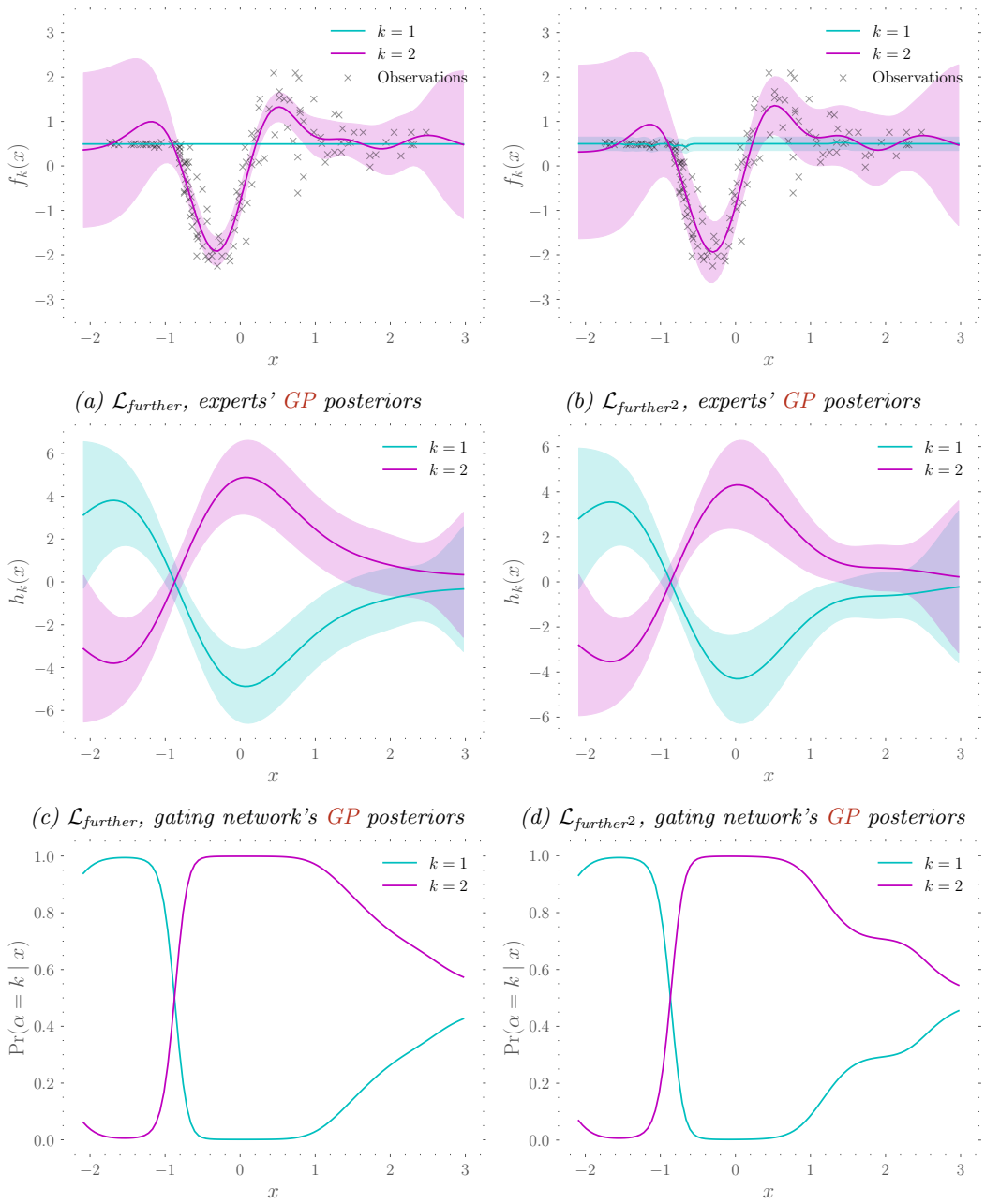
it is visualised by drawing samples from its posterior, i.e. sample a mode indicator variable α_* and then draw a sample from the corresponding expert.

Predictive posteriors Both MoSVGPE results are capable of modelling the non-stationarity at $x \approx -0.7$ better than the SVGP. At this non-stationary point, there are two modes in the MoSVGPE predictive distributions, indicated by the overlap in samples from each expert in Figures 3.3c and 3.3d. The SVGP has explained the

observations by increasing its single noise variance term. In contrast, both of the MoSVGPE results have been able to learn two noise variances and these reflect the noise in the observations much better. This is indicated by expert one learning a low noise variance and expert two a high noise variance (similar to the SVGP's noise variance).

Latent variables More insight into this behaviour can be obtained by considering the latent variables. Figure 3.4 shows the posteriors over the latent variables where Figures 3.4a and 3.4b show the GP posteriors over each expert's latent function $q(f_k(\mathbf{x}_*))$. Figures 3.4c and 3.4d show the GP posteriors over the latent gating functions $q(h_k(\mathbf{x}_*))$ and Figures 3.4e and 3.4f show the mixing probabilities associated with the probability mass function over the expert indicator variable α.

The lengthscale of the gating network kernel governs how fast the model can shift responsibility from expert one (cyan) to expert two (magenta). For both lower bounds, the distribution over the expert indicator variable tends to a uniform distribution (maximum entropy) at $x \geq 1.5$. This can be seen by the cyan/magenta lines in Figures 3.4e and 3.4f tending to 0.5. Optimising with both bounds resulted in expert one (cyan) learning a long lengthscale to fit the horizontal line from $-2 < x < -1$ and expert two (magenta) learning a shorter lengthscale function to fit the wiggly section from $-0.5 < x < 1.2$. The noise variance inferred by expert one is larger for $\mathcal{L}_{\text{further}^2}$ than for $\mathcal{L}_{\text{further}}$. The uncertainty in the experts' latent functions is also higher for $\mathcal{L}_{\text{further}^2}$. This is shown by the 95% confidence intervals (shaded cyan/-magenta) being wider in Figure 3.4b than in Figure 3.4a. This is because $\mathcal{L}_{\text{further}^2}$ is attempting to fit both experts to the entire data set and only mixes their noise models. In contrast, $\mathcal{L}_{\text{further}}$ fits each expert only in the regions where the gating network has assigned it responsibility.



(a) $\mathcal{L}_{further}$, experts' GP posteriors

(b) $\mathcal{L}_{further^2}$, experts' GP posteriors

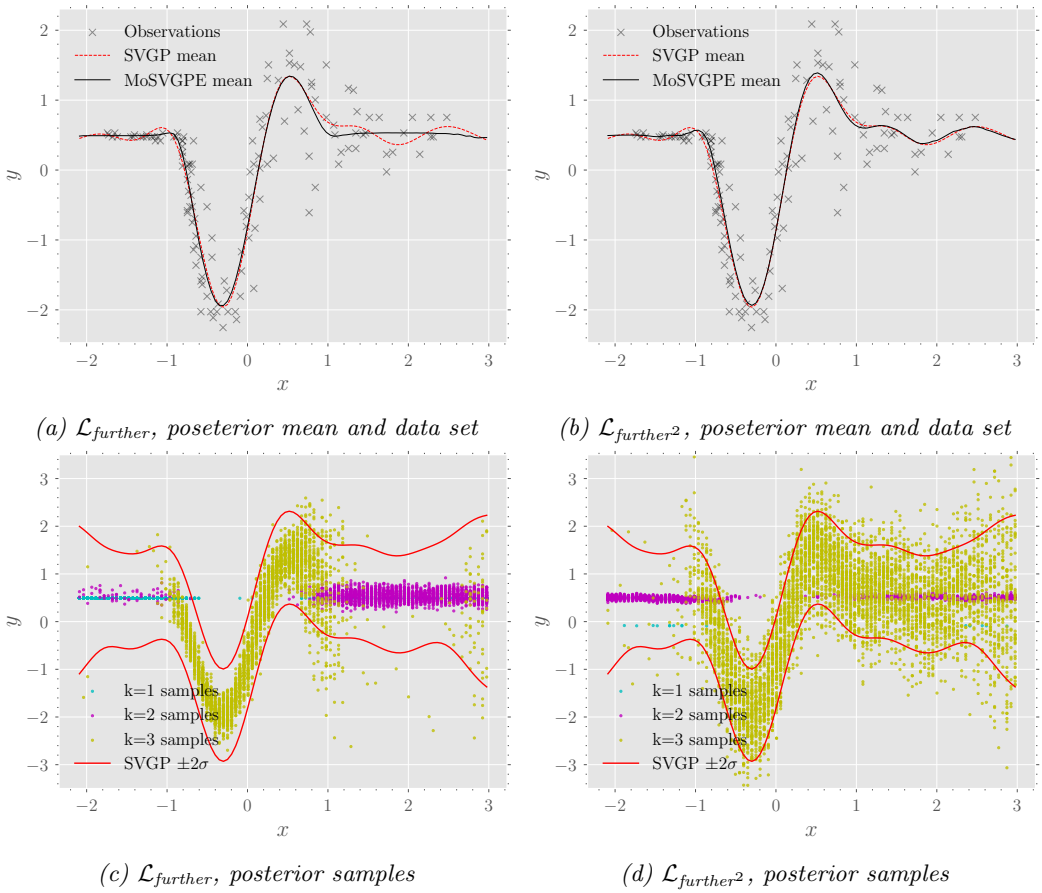
(c) $\mathcal{L}_{further}$, gating network's GP posteriors

(d) $\mathcal{L}_{further^2}$, gating network's GP posteriors

(e) $\mathcal{L}_{further}$, mixing probabilities

(f) $\mathcal{L}_{further^2}$, mixing probabilities

Figure 3.4: *Visualisation of the model's latent variables (instantiated with $K = 2$ experts), after training on the Motorcycle data set (Silverman, 1985), with $\mathcal{L}_{further}$ (left column) and with $\mathcal{L}_{further^2}$ (right column). (a-b) show the GP posteriors associated with the experts' latent functions $q(f_k(\mathbf{x}_*))$, where the solid lines show the mean and the shaded regions show the 95% confidence intervals, i.e. $\pm 2\sigma$. The gating network's GP posteriors $q(h_k(\mathbf{x}_*))$ are shown in (c-d) and their associated mixing probabilities $q(\alpha_n^* = k)$ in (e-f).*



(a) $\mathcal{L}_{further}$*, poseterior mean and data set*

(b) $\mathcal{L}_{further^2}$*, poseterior mean and data set*

(c) $\mathcal{L}_{further}$*, posterior samples*

(d) $\mathcal{L}_{further^2}$*, posterior samples*

Figure 3.5: *Visualisation of the model instantiated with* $K = 3$ *experts, after training on the motorcycle data set (Silverman, 1985) with* $\mathcal{L}_{further}$ *(left column) and with* $\mathcal{L}_{further^2}$ *(right column). (a-b) show the data set (black crosses) and the posterior means associated with the MoSVGPE (black solid line) and the SVGP (red dashed line) for comparison. (c-d) show samples from the MoSVGPE posterior where colour indicates the underlying expert (cyan* $K = 1$*, magenta* $K = 2$*, yellow* $K = 3$*) and the red lines show the* ± 2 *standard deviation error (95% confidence interval) of the SVGP posterior. All experiments were trained with* $25,000$ *iterations of Adam (Kingma and Ba, 2017) using a learning rate of* 0.01 *and a minibatch size of* $N_b = 16$*. All GPs use Squared Exponential kernels and* $M = 32$ *inducing points.*

Three Experts

The model was then instantiated with three experts $K = 3$ and trained following the same procedure as the two experts' experiments. Table 3.3 shows the initial values for all of the trainable parameters in the model. The results are shown in Figure 3.5, where the top row visualises the predictive mean and the bottom

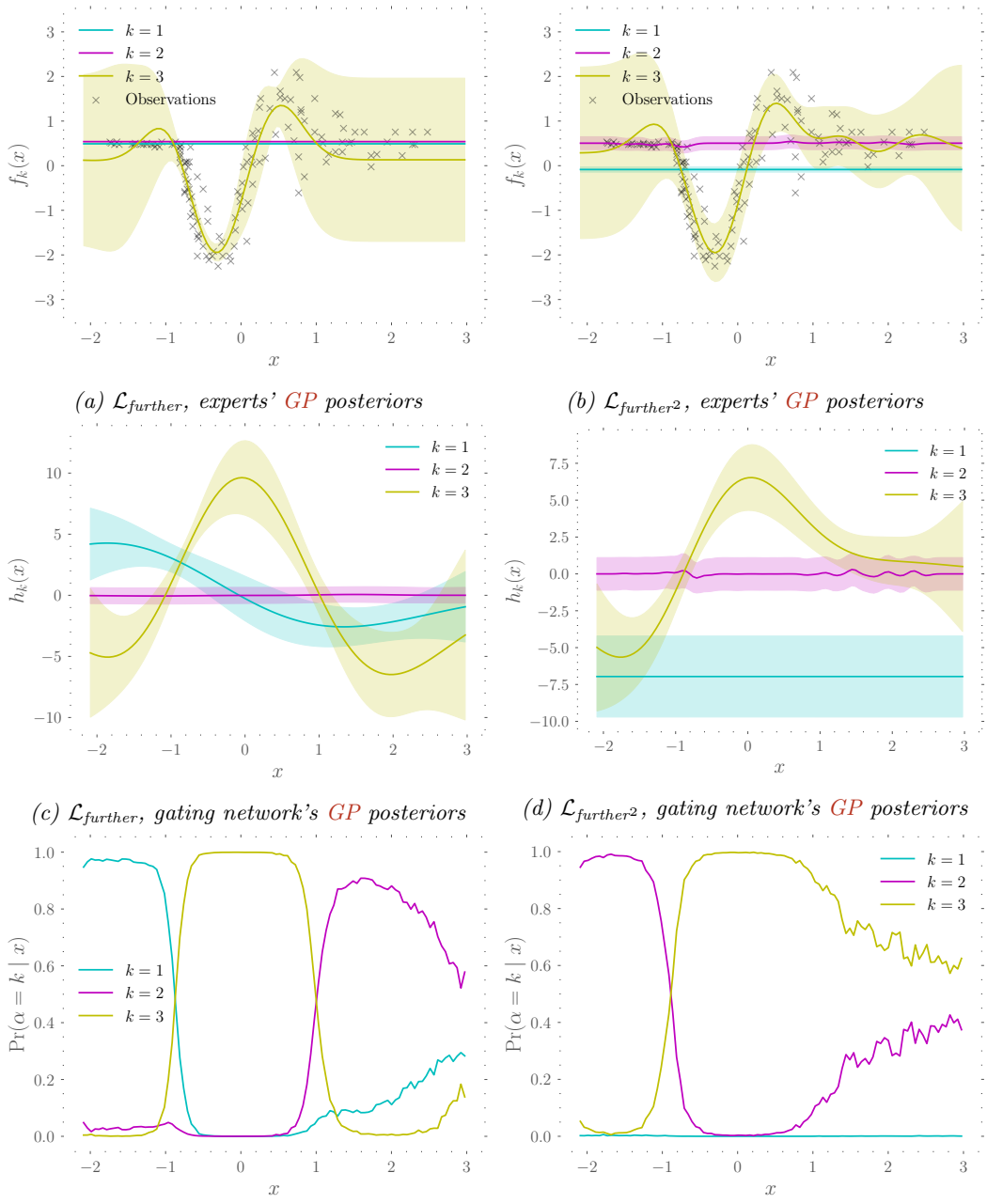
row the predictive density, for $\mathcal{L}_{\text{further}}$ (left column) and $\mathcal{L}_{\text{further}^2}$ (right column). Figure 3.6 then visualises the posteriors over the latent variables associated with each model/bound combination.

From Table 3.1, it is clear that the predictive posterior associated with $\mathcal{L}_{\text{further}}$ is the most accurate as it obtained the best NLPP score. As expected, the two lower bounds explain the data completely differently. Instantiating the model with three experts $K = 3$ and training with $\mathcal{L}_{\text{further}}$, leads to the extra expert (magenta) fitting to the data at $x \geq 1.5$ and the gating network assigning responsibility to it in this region. In contrast, instantiating the model with three experts $K = 3$ and training with $\mathcal{L}_{\text{further}^2}$, results in the gating network never using the extra expert (cyan). This is indicated by the extra expert's (cyan) probability remaining low over the region with training data in Figure 3.6f. Similar to the two expert case, the distribution over the expert indicator variable at $x \geq 1.5$ tends to a uniform distribution (maximum entropy), over the experts that are turned "on".

In Figure 3.6a the third expert's posterior returns to the prior at $x \geq 1.5$. This is indicated by the 95% confidence intervals associated with the third expert's GP (shaded yellow) being wide in this region. This demonstrates that not only is the gating network turning the experts "on" and "off" in different regions but the model is also exhibiting data assignment behaviour. That is, each expert appears to only be fitting to the observations in the regions where the gating network has assigned it responsibility. In our case, this behaviour is achieved via the inducing variables capturing the joint distribution over the experts and the set of assignments, i.e. implicitly assigning data points to experts.

Table 3.3: *Initial parameter settings before training on motorcycle data set with three experts.*

	Description	Symbol	Value
Optimiser	Num data	N	133
	Batch size	N_b	16
	Epochs	N/A	25000
	Num gating samples	$\hat{S}$	1
	Num expert samples	S	1
	Learning rate	N/A	0.01
Expert 1	Kernel variance	σ_f	0.1
	Kernel lengthscales	l	10
	Likelihood variance	σ_n	0.03
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\zeta}_1$	$\boldsymbol{\zeta}_1 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_1 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_1$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_1$	ones plus Gaussian noise
Expert 2	Kernel variance	σ_f	0.1
	Kernel lengthscales	l	10.0
	Likelihood variance	σ_n	0.1
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\zeta}_2$	$\boldsymbol{\zeta}_2 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_2 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_2$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_2$	ones plus Gaussian noise
Expert 3	Kernel variance	σ_f	1.0
	Kernel lengthscales	l	0.5
	Likelihood variance	σ_n	0.9
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\zeta}_3$	$\boldsymbol{\zeta}_3 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_3 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_3$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_3$	ones plus Gaussian noise
Gating functions (1-3)	Kernel variance	σ_f	3.0
	Kernel lengthscales	l	0.5
	Num inducing points	M	32
	Inducing inputs	$\boldsymbol{\xi}$	$\boldsymbol{\xi} \subseteq \mathbf{X}$ with $\#\boldsymbol{\xi} = M$
	Inducing variable mean	$\hat{\mathbf{m}}_k$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_k$	10× ones plus Gaussian noise



(a) $\mathcal{L}_{further}$*, experts' GP posteriors*

(b) $\mathcal{L}_{further^2}$*, experts' GP posteriors*

(c) $\mathcal{L}_{further}$*, gating network's GP posteriors*

(d) $\mathcal{L}_{further^2}$*, gating network's GP posteriors*

(e) $\mathcal{L}_{further}$*, mixing probabilities*

(f) $\mathcal{L}_{further^2}$*, mixing probabilities*

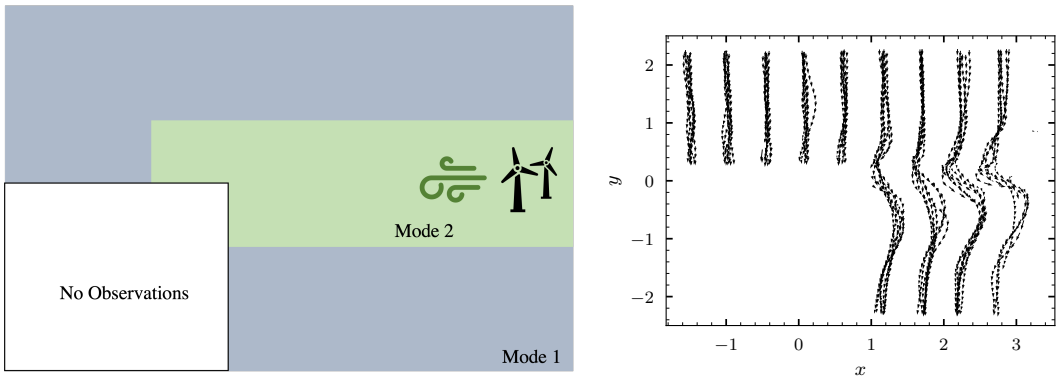
Figure 3.6: *Visualisation of the model's latent variables (instantiated with* $K = 3$ *experts), after training on the Motorcycle data set (Silverman, 1985), with* $\mathcal{L}_{further}$ *(left column) and with* $\mathcal{L}_{further^2}$ *(right column). (a-b) show the GP posteriors associated with the experts' latent functions* $q(f_k(\mathbf{x}_*))$*, where the solid lines show the mean and the shaded regions show the 95% confidence intervals, i.e.* $\pm 2\sigma$*. The gating network's GP posteriors* $q(h_k(\mathbf{x}_*))$ *are shown in (c-d) and their associated mixing probabilities* $q(\alpha_n^* = k)$ *in (e-f).*

Summary

The tight lower bound $\mathcal{L}_{\text{tight}}$ and further lower bound $\mathcal{L}_{\text{further}}$ recovered similar results in all experiments. This indicates that $\mathcal{L}_{\text{further}}$ does not loosen the bound to a point where it loses valuable information. In contrast, $\mathcal{L}_{\text{further}^2}$ is not able to recover the same results. This was expected as $\mathcal{L}_{\text{further}^2}$ corresponds to a further likelihood approximation, where the experts' noise models are mixed instead of their full SVGPs. $\mathcal{L}_{\text{further}}$ offers a rich ELBO for optimising MoSVGPE that achieves similar results to $\mathcal{L}_{\text{tight}}$, whilst having lower computational complexity per evaluation. For this reason, the remainder of this thesis uses $\mathcal{L}_{\text{further}}$ for all experiments.

3.5.3 Evaluation on Velocity Controlled Quadcopter

As this work is motivated by learning representations of real-world dynamical systems, it was tested on a real-world quadcopter data set following the illustrative example detailed in Section 1.1. The data set was collected at the Bristol Robotics Laboratory using a velocity controlled DJI Tello quadcopter and a Vicon tracking system. A high turbulence dynamics mode was induced by placing a desktop fan at the right side of a room. Figure 3.7a shows a diagram of the environment. The data set represents samples from a dynamical system with constant controls, i.e. $\Delta \mathbf{x}_{t+1} = f(\mathbf{x}_t; \mathbf{u}_t = \mathbf{u}_*)$.



(a) Diagram showing a top-down view of the environment (a room in the Bristol Robotics Laboratory (BRL)). *(b) Quiver plot showing the data set of state transitions from 9 trajectories flown 7 times.*

Figure 3.7: *Illustration of (a) the environment (a room in the Bristol Robotics Laboratory (BRL)) and (b) the data set of state transitions. A turbulent dynamics mode is induced by a desk top fan at the right-hand side of the room and a subset of the environment has not been observed.*

Environment The environment is modelled with two dimensions (the x and y coordinates), which is a realistic assumption, as altitude control can be achieved with a separate controller. The state space is then the 2D coordinates $\mathbf{x} = [x, y]$ and the control is simply the velocity $\mathbf{u} = [v_x, v_y]$.

Data collection The Vicon system provided access to the true position of the quadcopter at all times, which enabled pre-planned trajectories to be flown, using a simple PID controller on feedback from the Vicon system. To simplify data collec-

Table 3.4: *Initial parameter settings before training on the real-world velocity controlled quadcopter data set.*

	Description	Symbol	Value
Optimiser	Num data	N	1816
	Batch size	N_b	64
	Num epochs	N/A	10000
	Num gating samples	$\hat{S}$	1
	Num expert samples	S	1
	Learning rate	N/A	0.01
Expert 1	Constant mean function	c_1	$[0, 0]$
	Kernel variance	σ_f	0.1
	Kernel lengthscales	l	$[2, 2]$
	Likelihood variance	Σ_{ϵ_1}	$\text{diag}([0.0011, 0.0011])$
	Num inducing points	M	100
	Inducing inputs	$\boldsymbol{\zeta}_1$	$\boldsymbol{\zeta}_1 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_1 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_1$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_1$	ones plus Gaussian noise
Expert 2	Constant mean function	c_2	$[0, 0]$
	Kernel variance	σ_f	20
	Kernel lengthscales	l	$[0.5, 0.5]$
	Likelihood variance	Σ_{ϵ_2}	$\text{diag}([1.9, 1.9])$
	Num inducing points	M	100
	Inducing inputs	$\boldsymbol{\zeta}_2$	$\boldsymbol{\zeta}_2 \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_2 = M$
	Inducing variable mean	$\hat{\mathbf{m}}_2$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_2$	ones plus Gaussian noise
Gating function	Kernel variance	σ_f	0.6
	Kernel lengthscales	l	$[0.1, 0.1]$
	Num inducing points	M	100
	Inducing inputs	$\boldsymbol{\zeta}_k$	$\boldsymbol{\zeta}_k \subseteq \mathbf{X}$ with $\#\boldsymbol{\zeta}_k = M$
	Inducing variable mean	$\hat{\mathbf{m}}_k$	zeros plus Gaussian noise
	Inducing variable Cholesky	$\hat{\mathbf{S}}_k$	$2\times$ ones plus Gaussian noise

tion, nine trajectories from $y = 2$ to $y = -3$, with different initial x locations, were used as target trajectories to be tracked by the PID controller. Each trajectory was repeated 7 times to capture the variability (process noise) in the dynamics.

Data processing The Vicon stream recorded data at 100Hz, which was then down-sampled to give a time step of $\Delta t = 0.1s$. This reduced the size of the data set and left reasonable lengthscales. The data set consists of $N = 1816$ state transitions. Figure 3.7b visualises the state transition data set as a quiver plot.

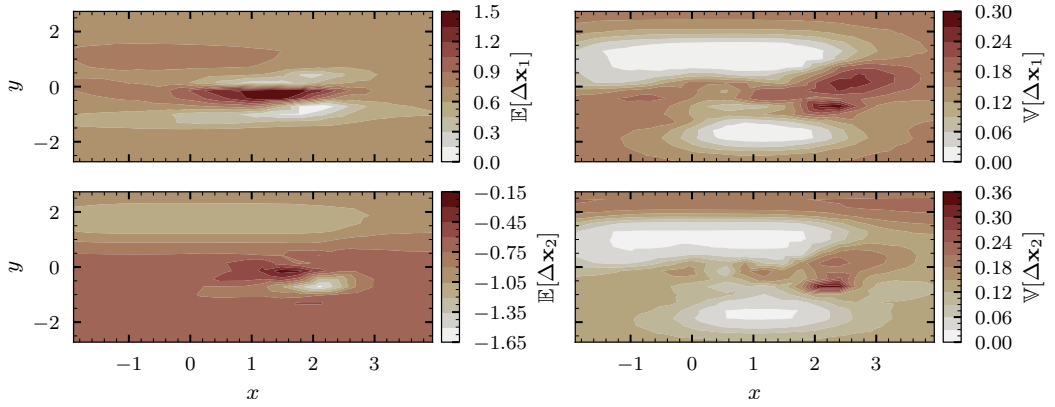
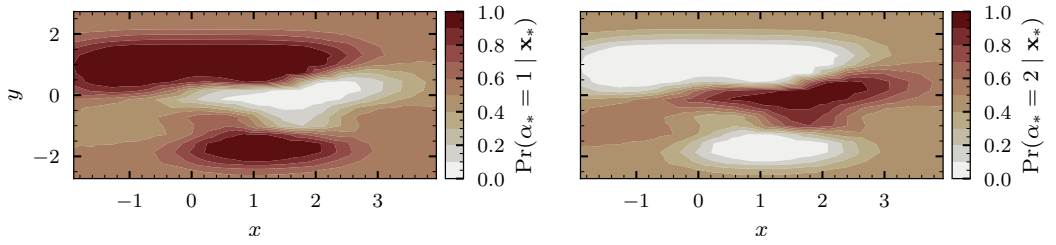


Figure 3.8: *Moment matched predictive posterior* $p(\Delta \mathbf{x}_* \mid \mathbf{x}_*)$ *after training on the quadcopter data set. Each row corresponds to an output dimension* d *where the left plot shows the moment matched mean* $\mathbb{E}[\Delta \mathbf{x}_d]$ *and the right plot shows the moment matched variance* $\mathbb{V}[\Delta \mathbf{x}_d]$.

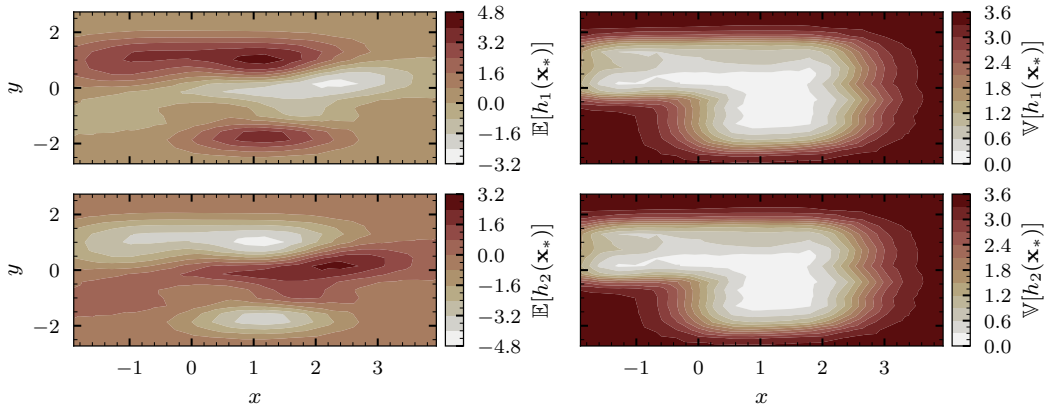
Results

The model was instantiated with two experts, with the goal of each expert learning a separate dynamics mode and the gating network learning a representation of how the underlying dynamics modes vary over the state space. The model was trained using the model and training parameters in Table 3.4.

At a new input location $\mathbf{x}_n^*$ the density over the output, $p(y_n^* \mid \mathbf{x}_n^*)$, follows a mixture of K Gaussians. Visualising a mixture of two Gaussians with a two-dimensional input space and a two-dimensional output space requires the components and mixing probabilities to be visualised separately. To aid with visualisation, Figure 3.8 shows the predictive density approximated as a unimodal Gaussian density (via moment matching), where each row corresponds to an output dimension. The predictive mean is fairly constant over the domain, except for the region in front of the fan, where it is higher. This result makes sense as the data set was assumed to be collected with constant controls. The region with high predictive mean in front of the fan is modelling the drift arising from the fan blowing the quadcopter in the negative x direction. The right-hand plots of Figure 3.8 show the predictive variance. It is high in the bottom left where there are no training observations, indicating that



(a) Posterior over mode indicator variable.



(b) GP posteriors over gating functions.

Figure 3.9: *Visualisation of the gating network after training on the quadcopter data set. The plots in (a) show the predictive mixing probabilities* $\Pr(\alpha_n^* = k \mid \mathbf{y})$ *for Expert 1 (left) and Expert 2 (right). The plots in (b) show the predictive GP posteriors* $q(h_k(\mathbf{x}_n^*))$ *associated with Expert 1 (top) and Expert 2 (bottom). The left-hand plots show the means and the right-hand plots show the variances.*

the method has successfully represented the model's *epistemic uncertainty*. It is also high in the region in front of the fan, showing that the model has successfully inferred the high process noise, associated with the turbulence induced by the fan. The individual experts and the gating network are now visualised separately.

Gating network Figure 3.9 shows the gating network after training on the data set. Figure 3.9a (right) indicates that the model has assigned responsibility to Expert 2 in front of the fan, as its mixing probability $\Pr(\alpha_n^* = 1 \mid \mathbf{x}_n^*)$ is high (red) in this region. This implies that Expert 2 represents the turbulent dynamics mode in front of the fan. Figure 3.9b shows the GP posteriors associated with the gating functions. The mean of the gating function associated with Expert 1 $\mathbb{E}[h_1(\mathbf{x}_n^*)]$ is high (red) in the low-turbulence regions and low (white) in the high-turbulence region in front of

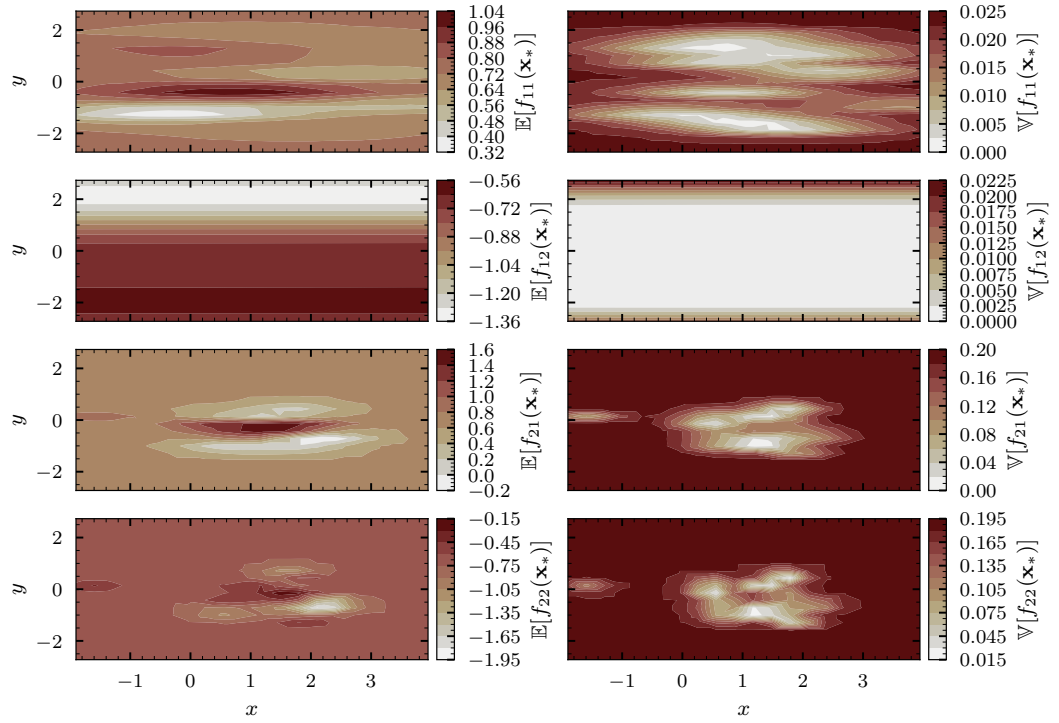


Figure 3.10: *Visualisation of the experts' predictive posteriors after training on the quadcopter data set. Each row corresponds to a single GP posterior, $q(f_{kd}(\mathbf{x}_n^*))$, corresponding to dimension d of expert k. The mean $\mathbb{E}[f_{kd}(\mathbf{x}_n^*)]$ is on the left and the variance $\mathbb{V}[f_{kd}(\mathbf{x}_*)]$ is on the right. The noise variances learned by Expert 1 and Expert 2 were $\Sigma_1 = \mathrm{diag}([0.0063, 0.0259])$ and $\Sigma_2 = \mathrm{diag}([0.0874, 0.0432])$ respectively.*

the fan. The posterior variance associated with the gating function GPs is high in the region with no training observations. This is a desirable behaviour because it is modelling the *epistemic uncertainty*. These results demonstrate that the gating network infers important information regarding how the system switches between dynamics modes over the input space.

Identifiability These results show that the GP-based gating network is capable of turning a single expert on in multiple regions of the input space. This is a desirable behaviour as it has enabled only two underlying dynamics modes to be identified. In contrast, other MoGPE methods may have assigned an extra expert to one of the regions modelled by Expert 1. In particular, the regions at $y > 0$ and $y < -1$ may have been assigned to separate experts.

Experts Figure 3.10 shows the predictive posteriors $q(f_{kd}(\mathbf{x}_n^*))$ associated with each dimension d of each expert k. The method has successfully learned a factorised representation of the underlying dynamics, where Expert 1 has learned a dynamics mode with low process noise $\Sigma_1 = \text{diag}([0.0063, 0.0259])$ and Expert 2 a mode with high process noise $\Sigma_2 = \text{diag}([0.0874, 0.0432])$. Expert 2 has also clearly learned the drift induced by the fan, indicated by the dark red region at $y = 0$ in the two bottom left plots of Figure 3.10. It has also learned the control response of the PID controller correcting for the deviation from the reference trajectory, indicated by the white region below $y = 0$. The control response is an artefact of the data collection process. Expert 2 has therefore learned both the drift and process noise terms associated with the turbulent dynamics mode.

Both experts were initialised with independent inducing inputs, $\boldsymbol{\zeta}_k$, providing the model flexibility to "soft" partition the data set. That is, each expert has the freedom to set its inducing inputs, $\boldsymbol{\zeta}_k$, to support only a subset of the data set. The posterior (co)variance associated with each expert represents their *epistemic uncertainty*. The top right plot in Figure 3.10 shows the posterior variance associated with the x output dimension of Expert 1. The posterior variance increases in front of the fan because the gating network has assigned responsibility to the other expert in this region. However, the posterior variance associated with the y output dimension of Expert 1, is not high in this region. This is due to the lengthscale of the y output dimension allowing Expert 1 to confidently extrapolate.

The bottom right two plots in Figure 3.10 show the posterior variance associated with the x and y output dimensions of Expert 2. The posterior variance is high everywhere except for the region in front of the fan. Again, this is due to the gating network assigning responsibility to the other expert outside of the region in front of the fan. These results demonstrate that the likelihood approximation in Equation (3.25), combined with our gating network and variational inference scheme, is capable of modelling the assignment of observations to experts via the inducing points.

3.6 Discussion and Future Work

Implicit data assignment It is worth noting that in contrast to other MoGPE methods, this model does not directly assign observations to experts. However, after augmenting each expert with separate inducing points, the model has the flexibility to loosely *partition* the data set. Just as sparse GP methods can be viewed as methods that parameterise the full nonparametric GP, our approach can be viewed as parameterising the nonparametric MoGPE. Conveniently, our parameterisation, in particular the likelihood approximation in Equation (3.25), deals with the issue of marginalising exponentially many sets of assignments of observations to experts. As evident from the results in this chapter, this likelihood approximation appears to retain important information regarding the assignment of observations to experts, whilst efficiently marginalising the expert indicator variable. It is also worth noting that the number of inducing points M associated with each expert, could be set by considering the number of data points believed to belong to a particular expert. Currently, each expert's inducing inputs are initialised by randomly sampling a subset of the data inputs. Future work could explore different techniques for initialising each expert's inducing inputs.

Bayesian treatment of inducing inputs Common practice in sparse GP methods is to jointly optimise the hyperparameters and the inducing inputs. Optimising only some of the parameters, instead of marginalising all of them, is known as Type-II maximum likelihood. In Bayesian model selection, it is well-known that Type-II maximum likelihood can lead to overfitting if the number of parameters being optimised is large. In the case of inducing inputs, there can often be beyond hundreds or thousands that need to be optimised. Further to this, Rossi et al., 2021 show that optimising the inducing inputs relies on being able to optimise both the prior and the posterior, therefore contradicting Bayesian inference. Our variational inference scheme follows common practice and optimises the inducing inputs jointly with the hyperparameters. In some instances, we observe that optimising the inducing

inputs leads to them taking values far away from the training data. Often this can be avoided by simply sampling the inducing inputs the training inputs and fixing them, i.e. not optimising them. This often leads to better NLPP scores as well. This observation highlights that a Bayesian treatment of the inducing inputs is an interesting direction for future work. However, specifying priors and performing *efficient* posterior inference over the inducing inputs is a challenging problem.

Latent spaces for control The gating network consists of two spaces which are rich with information regarding how the system switches between it's underlying dynamics modes, namely, the pmf over the expert indicator variable and the GP posteriors over the gating functions. It is worth noting that all MoGPE methods have a pmf over the expert indicator variable. However, this space suffers from interpretability issues. This is because in conventional MoGPE methods, the *epistemic uncertainty* associated with the gating network is not decoupled from the pmf over the expert indicator variable. Consider the meaning of the mixing probabilities tending to a uniform distribution $\Pr(\alpha_n^* = k \mid \mathbf{x}_n^*) = 0.5$. This corresponds to maximum entropy for a categorical distribution and could mean two different things. It could mean that,

1. It has **high** *epistemic uncertainty*, so cannot confidently predict which expert is responsible,
2. It has **low** *epistemic uncertainty* and confidently mixes the experts' predictions,
    - This happens at the boundaries between experts.

This interpretability issue is overcome by our GP-based gating network, as these two cases are modelled differently. Either the gating function(s) are all equal and their posterior variance(s) are low, implying that the gating network has **low** epistemic uncertainty and is likely at a boundary between experts. Alternatively, the gating functions' posterior variance(s) could be high, implying it has **high** *epistemic uncertainty*.

Importantly, the GP posteriors associated with our gating network, not only infer information regarding the mode switching but also model the gating network's *epistemic uncertainty*. Further to this, formulating the gating network GPs with differentiable mean and covariance functions, enables techniques from Riemannian geometry to be deployed on the gating functions (Carmo, 1992). The power of the GP-based gating network will become apparent when its latent *geometry* is leveraged for control in Section 4.2 and when its GPs are used to develop an information-based exploration strategy in Chapter 6.

3.7 Conclusion

This chapter has presented a method for learning representations of multimodal dynamical systems using a MoGPE method. Motivated by correctly identifying the underlying dynamics modes and inferring latent structure that can be exploited for control, this work formulated a gating network based on input-dependent gating functions. This aids the inherent identifiability issues associated with mixture models as it can be used to constrain the set of admissible functions through the placement of informative GP priors on the gating functions. Further to this, the GP posteriors over the gating functions provide convenient latent spaces for control. This is because they are rich with information regarding the separation of the underlying dynamics modes and also model the *epistemic uncertainty* associated with the gating network. In later chapters this uncertainty will be used to construct risk-averse control strategies and to guide exploration for MBRL.

The variational inference scheme presented in this chapter addresses the issue of marginalising every possible set of assignments of observations to experts – of which there are K^N possibilities – in the MoGPE marginal likelihood. It overcomes the issue of assigning observations to experts by augmenting each expert GP with a set of inducing points. These inducing points are assumed to be a sufficient statistic for the joint distribution over every possible set of assignments to experts. This

induces a factorisation over data which is used to derive three ELBOs that provide a coupling between the optimisation of the experts and the gating network, by efficiently marginalising the expert indicator variable for single data points. The ELBOs are compared on the Motorcycle data set (Silverman, 1985). The $\mathcal{L}_{\text{further}}$ bound provides the best performance as it balances the accuracy offered by the tight bound $\mathcal{L}_{\text{tight}}$, with the computational improvements offered by further bounding the GPs. The results demonstrate that the variational inference scheme principally handles uncertainty whilst providing scalability via stochastic variational inference. The method is further evaluated on a real-world quadcopter example demonstrating that it can successfully learn a factorised representation of a real-world, multimodal, robotic system.

4 Mode Remaining Trajectory Optimisation

This chapter is concerned with controlling *unknown* or *partially unknown*, multi-modal dynamical systems, given a single-step predictive dynamics model learned using the MoSVGPE method from Chapter 3. In particular, it is concerned with *mode remaining* trajectory optimisation, which is formally defined in Definition 2.1.1. Informally, *mode remaining* trajectory optimisation attempts to find trajectories from an initial state $\mathbf{x}_0$ – in the desired dynamics mode – to a target state $\mathbf{x}_f$, whilst remaining in the desired dynamics mode.

The MoSVGPE method from Chapter 3 was intentionally formulated with latent variables – to represent the mode switching behaviour and its associated uncertainty – so that they could be leveraged to encode mode remaining behaviour into control strategies. This chapter unleashes the power of these latent variables by making decisions under their uncertainty.

The remainder of this chapter is organised as follows. Section 4.1 formally states the problem. Section 4.2 details two methods that leverage the geometry of the MoSVGPE gating network. The first method in Section 4.2.2 resembles an indirect optimal control method as it solves the necessary conditions which *indirectly* represent the original optimal control problem. In contrast, the second method in Section 4.2.3 takes a more standard approach and directly solves the optimal control problem. Section 4.3 then introduces an alternative approach to mode remaining trajectory optimisation, which does not leverage the geometry of the gating network.

Instead, it extends the Control as Inference (CaI) framework (Toussaint, 2009) and encodes mode remaining behaviour via conditioning on the mode indicator variable. Chapter 5 evaluates and compares all three methods using the illustrative example from Section 1.1. An initial version of the Indirect Optimal Control via Latent Geodesics (IG) method presented in Section 4.2.2 is published in Scannell et al., 2021.

4.1 Problem Statement

The goal of this chapter is to solve the mode remaining navigation problem in Section 2.1. Due to the novelty of this problem, the work in this chapter considers trajectory optimisation algorithms rather than state feedback (closed-loop) controllers. This mode remaining trajectory optimisation problem is given by,

$$\min_{\bar{\mathbf{u}}} \sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \tag{4.1a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f_k(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon}_k \quad \text{if} \quad \alpha(\mathbf{x}_t) = k \qquad \forall t \in \{0, \dots, T-1\} \tag{4.1b}$$

$$f(\mathbf{x}_t, \mathbf{u}_t) \in \mathcal{X}_{k^*} \qquad \forall t \in \{0, \dots, T-1\} \tag{4.1c}$$

$$\mathbf{u}_t \in \mathcal{U} \qquad \forall t \in \{0, \dots, T-1\} \tag{4.1d}$$

$$\mathbf{x}_0 = \mathbf{x}_0 \tag{4.1e}$$

$$\mathbf{x}_T = \mathbf{x}_f, \tag{4.1f}$$

where the dynamics are from Equation (2.1) and the resulting open-loop controller is given by, $\pi(t) = \mathbf{u}_t \quad \forall t \in \{0, \dots, T-1\}$. Given the desired dynamics mode k^*, Equation (4.1) seeks to find a control trajectory $\bar{\mathbf{u}} = \mathbf{u}_{0:T-1}$, to navigate from an initial state $\mathbf{x}_0 \in \mathcal{X}_{k^*}$, to a target state $\mathbf{x}_f \in \mathcal{X}_{k^*}$, over a horizon T, whilst minimising a cost function, $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ and keeping the system in the desired dynamics mode k^*. To simplify notation, the state and control trajectories are denoted as $\bar{\mathbf{x}} = \mathbf{x}_{1:T}$ and $\bar{\mathbf{u}} = \mathbf{u}_{0:T-1}$ respectively.

Given that neither the underlying dynamics modes nor how the system switches between them, are *known a priori*, it is not possible to solve Equation (4.1) with the mode remaining guarantee in Definition 2.1.1. However, well-calibrated uncertainty estimates associated with a learned dynamics model make it possible to find mode remaining trajectories with high probability. Therefore, this work relaxes the requirement to finding mode remaining trajectories with high probability. Let us formally define a $\delta-$mode remaining controller π.

Definition 4.1.1 (δ-mode remaining)**.** *Let $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ denote a multimodal dynamical system and k^* a desired dynamics mode defined by its state domain $\mathcal{X}_{k^*} = \{\mathbf{x} \in \mathcal{X} \mid \alpha(\mathbf{x}) = k^*\}$. Given an initial state $\mathbf{x}_0 \in \mathcal{X}_{k^*}$ and $\delta \in (0, 1]$, a controlled system is said to be δ-mode remaining under the controller $\pi \in \Pi$ iff:*

$$\Pr(\forall t \in \{0, \dots, T-1\} : f(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \in \mathcal{X}_{k^*}, \pi(\mathbf{x}_t, t) \in \mathcal{U}) \geq 1 - \delta \tag{4.2}$$

Trajectories satisfying this $\delta-$mode remaining definition are guaranteed to remain in the desired dynamics mode with probability up to $1-\delta$. Therefore, smaller δ values correspond to a higher confidence of remaining in the desired dynamics mode.

This chapter assumes prior access to the environment, such that a data set of state transitions has previously been collected and used to learn a single-step dynamics model.

Assumption 4.1.1. *A data set $\mathcal{D}$ of state transitions has previously been collected from the system and used to learn a single-step dynamics model using the MoSVGPE method from Chapter 3.*

Given this learned dynamics model, it is assumed that a desired dynamics mode k^* is either known or can easily be identified. This is a realistic assumption as the parameters associated with each dynamics GP can be used to identify different behaviours. For example, the noise variance associated with each mode's dynamics GP models its process noise. Therefore, it is easy to identify undesirable dynamics modes with high process noise.

Assumption 4.1.2. *A desired dynamics mode k^* is known.*

Given a learned dynamics model and a desired dynamics mode k^*, the goals of the trajectory optimisation in this chapter can be summarised as follows,

Goal 1 Navigate to the target state $\mathbf{x}_f$,

Goal 2 Remain in the operable, desired dynamics mode k^*,

Goal 3 Avoid regions of the learned dynamics with high *epistemic uncertainty*,

- **Goal 3.1** in the desired dynamics mode f_{k^*}, i.e. where the underlying dynamics are not known,
- **Goal 3.2** in the gating network α, i.e. where it is not known which mode governs the dynamics.

Goal 3 arises due to learning the dynamics model from observations. The learned model may not be able to confidently predict which mode governs the dynamics in a given region. This is due to a lack of training observations and is known as *epistemic uncertainty*. It is desirable to avoid entering these regions as it may result in the system leaving the desired dynamics mode.

4.2 Mode Remaining Control via Latent Geometry

This section introduces two different approaches to performing mode remaining trajectory optimisation. They both exploit concepts from Riemannian geometry – extended to probabilistic manifolds – to encode mode remaining behaviour. The first approach in Section 4.2.2 resembles an indirect optimal control method (Kirk, 2004) as it projects the trajectory optimisation problem onto an Ordinary Differential Equation (ODE) that implicitly encodes the mode remaining behaviour. As such, we name this approach Indirect Optimal Control via Latent Geodesics (IG). The second approach in Section 4.2.2 is a direct optimal control method that resembles standard Gaussian process control methods with the mode remaining behaviour encoded via a geometric objective function. We name this approach Direct Optimal Control via Riemannian Energy (DRE).

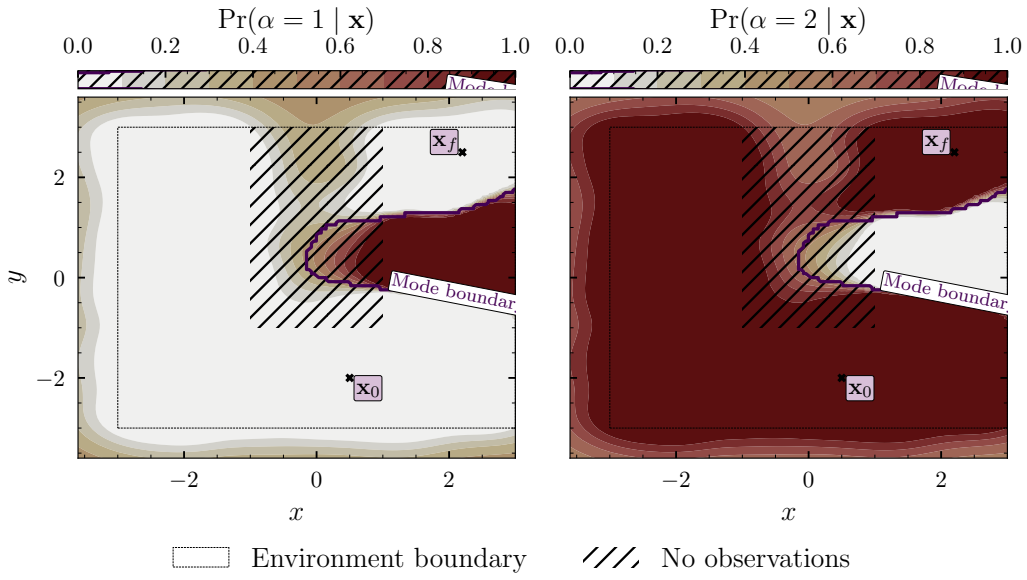
4.2.1 Concepts from Riemannian Geometry

The MoSVGPE model correctly identifies the underlying dynamics modes and infers informative latent spaces that can be used to encode mode remaining behaviour. Figure 4.1 shows the gating network posterior after training MoSVGPE on the historical data set of state transitions from the illustrative quadcopter example in Section 1.1. The work in this chapter is based on the observation that Goals 1 and 2 can be encoded as finding length minimising trajectories on the manifold parameterised by the desired mode's gating function, shown in the left-hand plot of Figure 4.1b. Intuitively, the length of a trajectory from $\mathbf{x}_0$ to $\mathbf{x}_f$ on the manifold given by the desired mode's gating function, increases when it passes over the contours; analogous to climbing a hill. Given appropriate scaling of the gating function, shortest trajectories between two locations are those that attempt to follow the contours, and as a result, remain in a single mode by not climbing up or down any hills. This section will review the relevant concepts from Riemannian geometry and show how they can be used to encode Goals 1 and 2. Section 4.2.1 then extends these concepts to probabilistic geometries to encode Goal 3.

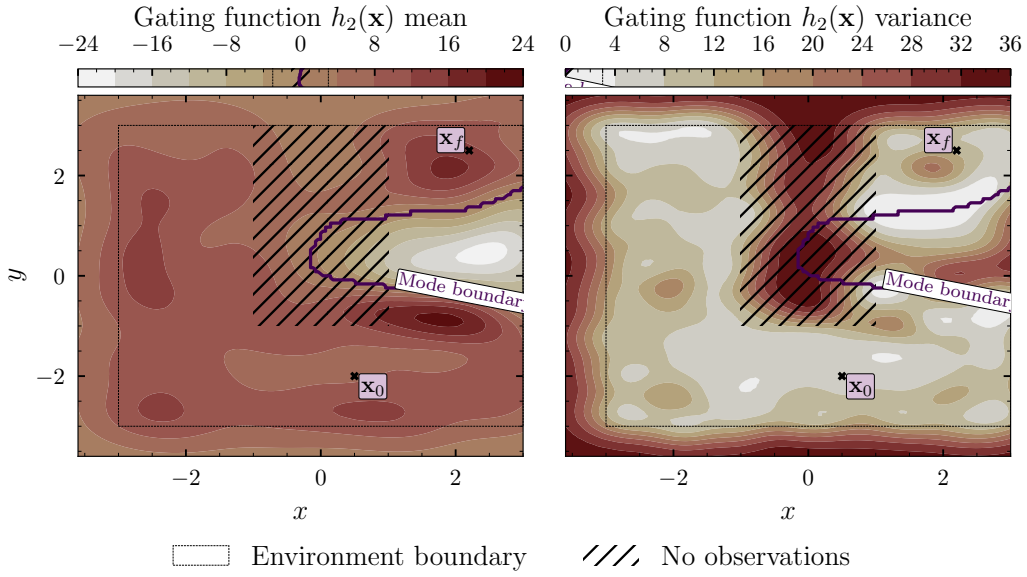
Lengths in Euclidean spaces The l^2 norm (Euclidean norm) provides an intuitive notion for the length of a vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ in a Euclidean space. A continuous-time trajectory is denoted $\bar{\mathbf{x}} : [t_0, t_f] \to \mathcal{X}$. Note that this has overloaded the discrete-time trajectory notation. Under the l^2 norm, the length of a trajectory $\bar{\mathbf{x}}$ is given by,

$$\text{Length}(\bar{\mathbf{x}}) = \int_{t_0}^{t_f} \|\dot{\mathbf{x}}(t)\|_2 \mathrm{d}t, \tag{4.3}$$

where Newton's notation has been used to denote differentiation with respect to time t. As a norm can be expressed for any space endowed with an inner product, it is possible to calculate lengths of trajectories on manifolds endowed with an inner product.



(a) Mixing probabilities



(b) Gating function's GP posterior

Figure 4.1: *Visualisation of the gating network posterior after training MoSVGPE on the state transition data set from the simulated version of the 2D quadcopter environment in the illustrative example from Section 1.1. (a) shows the probability mass function over the expert indicator variable and (b) shows the gating function's GP posterior mean (left) and posterior variance (right). The start state* $\mathbf{x}_0$ *and target state* $\mathbf{x}_f$ *are overlayed along with the mode boundary (purple line) and the subset of the environment which has not been observed (hashed box).*

Riemannian manifolds In this thesis it suffices to consider manifolds $\mathcal{M}$ defined by a mapping,

$$h : \mathcal{X} \to \mathcal{Z}, \tag{4.4}$$

where $\mathcal{X}$ and $\mathcal{Z}$ are open subsets of Euclidean spaces. The manifold $\mathcal{M}$ is given by $\mathcal{M} = h(\mathcal{X})$ and is said to be immersed in the ambient space $\mathcal{Z}$. The dimensionality of the surface is denoted $d_{\mathcal{X}} = \dim(\mathcal{X})$ whilst $d_{\mathcal{Z}} = \dim(\mathcal{Z})$ denotes the dimensionality of the ambient space. Riemannian manifolds can intuitively be seen as $d_{\mathcal{Z}}$-dimensional curved surfaces with a smoothly varying positive-definite inner product, governed by the Riemannian metric $\mathbf{G}$ (Carmo, 1992).

Definition 4.2.1 (Riemannian Metric)**.** *A Riemannian metric* $\mathbf{G}$*, on a manifold* $\mathcal{M}$*, is a smooth function* $\mathbf{G} : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{X}} \times d_{\mathcal{X}}}$ *that assigns a symmetric positive definite matrix to any point in* $\mathcal{X}$*.*

Intuitively, the metric forms a local inner product in $\mathcal{X}$ that informs how to measure lengths on the manifold $\mathcal{M}$, locally in $\mathcal{X}$. This is indicated in Equation (4.8). Riemannian manifolds locally resemble Euclidean spaces and have globally defined differentiable structure.

Lengths on Riemannian manifolds The length of a trajectory $\bar{\mathbf{x}}$ on a manifold $\mathcal{M}$, can be calculated by mapping it through the function h and using Equation (4.3),

$$\text{Length}(h(\bar{\mathbf{x}})) = \int_{t_0}^{t_f} \left\| \dot{h}(\mathbf{x}(t)) \right\|_2 \mathrm{d}t. \tag{4.5}$$

Applying the chain-rule allows Equation (4.5) to be expressed in terms of the Jacobian and the velocity,

$$\text{Length}(h(\bar{\mathbf{x}})) = \int_{t_0}^{t_f} \|\mathbf{J}(\mathbf{x}(t))\dot{\mathbf{x}}(t)\|_2 \mathrm{d}t, \tag{4.6}$$

$$\mathbf{J}(\mathbf{x}(t)) = \frac{\partial h}{\partial \mathbf{x}(t)} \in \mathbb{R}^{1\times d_{\mathcal{X}}}. \tag{4.7}$$

This implies that the length of a trajectory on the manifold $\mathcal{M}$, can be calculated in the input space $\mathcal{X}$, using a locally defined norm,

$$\begin{aligned}\|\mathbf{J}(\mathbf{x}(t))\dot{\mathbf{x}}(t)\|_2 &= \sqrt{(\mathbf{J}(\mathbf{x}(t))\dot{\mathbf{x}}(t))^T(\mathbf{J}(\mathbf{x}(t))\dot{\mathbf{x}}(t))} \\ &= \sqrt{\dot{\mathbf{x}}^T(t)\mathbf{G}_{\mathbf{x}_t}\dot{\mathbf{x}}(t)} := \|\dot{\mathbf{x}}(t)\|_{\mathbf{G}_{\mathbf{x}_t}},\end{aligned} \tag{4.8}$$

where $\mathbf{G}_{\mathbf{x}_t} = \mathbf{J}(\mathbf{x}(t))^T\mathbf{J}(\mathbf{x}(t))$ is a symmetric positive definite matrix (akin to a local Mahalanobis distance measure), known as the natural Riemannian metric. The length of a trajectory on a manifold $\mathcal{M}$, endowed with the metric $\mathbf{G}$, can then be calculated with,

$$\text{Length}(h(\bar{\mathbf{x}})) = \int_{t_0}^{t_f} \|\dot{\mathbf{x}}(t)\|_{\mathbf{G}_{\mathbf{x}_t}} \mathrm{d}t. \tag{4.9}$$

Figure 4.1 shows the GP posterior over the desired mode's gating function, $h_{k^*} : \mathcal{X} \to \mathcal{Z}$. Consider finding length minimising trajectories on the manifold $\mathcal{M}_{k^*} = h_{k^*}(\mathcal{X})$ associated with the desired mode's gating function, where the metric is given by,

$$\mathbf{G} = \mathbf{J}^T\mathbf{J}, \tag{4.10}$$

$$\mathbf{J} = \frac{\partial h_{k^*}}{\partial \mathbf{x}(t)} \in \mathbb{R}^{1\times D}. \tag{4.11}$$

These trajectories will attempt to remain in the desired dynamics mode k^*, encoding Goals 1 and 2. However, length minimising trajectories subject to this metric do not encode Goal 3. That is, they will not avoid regions of the learned dynamics, which cannot be predicted confidently due to high *epistemic uncertainty*. Goal 3 can

be encoded by observing that the metric tensor is actually a random variable and extending the concepts of length minimising trajectories to probabilistic manifolds.

Probabilistic Geometries

Following Tosi et al., 2014 we formulate a metric tensor that captures the variance in the manifold via a probability distribution. First note that as the differential operator is linear, the derivative of a GP is also a GP, assuming that the mean and covariance functions are differentiable.

Assumption 4.2.1 (Differentiable Gaussian Process)**.** *Let* $\mu : \mathcal{X} \to \mathbb{R}$ *and* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *denote the mean and covariance functions associated with a Gaussian process. The Gaussian process is differentiable iff* $\exists \frac{\partial \mu(\mathbf{x})}{\partial \mathbf{x}}, \frac{\partial^2 k(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x} \partial \mathbf{x}'} \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

Gaussian Process Jacobian As the differential operator is linear, a function $h : \mathcal{X} \to \mathbb{R}$ distributed as a GP,

$$h(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot, \cdot)), \tag{4.12}$$

where μ and k represent the mean and covariance functions, is jointly Gaussian with its Jacobian at a new input location $\mathbf{x}_* \in \mathbb{R}^{1 \times D}$,

$$\mathbf{J}_* = \mathbf{J}(\mathbf{x}_*) = \frac{\partial h}{\partial \mathbf{x}_*} \in \mathbb{R}^D, \tag{4.13}$$

assuming that the mean and covariance functions are differentiable. As such, the conditional distribution over the Jacobian $\mathbf{J}_*$ can be obtained using the properties of multivariate normals and is given by,

$$\mathbf{J}_* \sim \mathcal{N}\left(\underbrace{\frac{\partial \mu}{\partial \mathbf{x}_*} + \partial \mathbf{K}_{*N} \mathbf{K}_{NN}^{-1} h(\mathbf{X})}_{\boldsymbol{\mu}_\mathbf{J}}, \underbrace{\partial^2 \mathbf{K}_{**} - \partial \mathbf{K}_{*N} \mathbf{K}_{NN}^{-1} \partial \mathbf{K}_{N*}}_{\boldsymbol{\Sigma}_\mathbf{J}} \right), \tag{4.14}$$

where the covariance matrices are given by,

$$\mathbf{K}_{NN} = k(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{N \times N} \tag{4.15}$$

$$\partial \mathbf{K}_{*N} = \frac{\partial k(\mathbf{x}_*, \mathbf{X})}{\partial \mathbf{x}_*} \in \mathbb{R}^{D \times N} \tag{4.16}$$

$$\partial^2 \mathbf{K}_{**} = \frac{\partial^2 k(\mathbf{x}_*, \mathbf{x}_*)}{\partial \mathbf{x}_* \partial \mathbf{x}_*} \in \mathbb{R}^{D \times D}\,. \tag{4.17}$$

Equation (4.14) is a D-dimensional multivariate normal distribution.

Therefore, the metric tensor $\mathbf{G}$ in Equation (4.10) is the outer product of two normally distributed random variables. As such, the metric tensor $\mathbf{G}$ is also a random variable, following a non-central Wishart distribution (Anderson, 1946),

$$\mathbf{G} \sim \mathcal{W}_D\big(P, \boldsymbol{\Sigma}_{\mathbf{J}}, \mathbb{E}\big[\mathbf{J}^T\big]\mathbb{E}[\mathbf{J}]\big), \tag{4.18}$$

where P is the number of degrees of freedom (always one in our case) and $\mathbb{E}[\mathbf{J}]$ and $\boldsymbol{\Sigma}_{\mathbf{J}}$ are the mean and covariance matrices associated with the GP over the Jacobian. The expected value of the metric tensor in Equation (4.18) is given by,

$$\mathbb{E}[\mathbf{G}] = \mathbb{E}[\mathbf{J}^T]\,\mathbb{E}[\mathbf{J}] + \boldsymbol{\Sigma}_{\mathbf{J}}. \tag{4.19}$$

Importantly, this expected metric tensor includes a covariance term $\boldsymbol{\Sigma}_{\mathbf{J}}$, which implies that lengths on the manifold calculated under this expected metric will increase in areas of high covariance. This is a desirable behaviour because it encourages length minimising trajectories to avoid regions of the learned dynamics with high *epistemic uncertainty*, encoding Goal 3. To aid with user control, the metric tensor in Equation (4.19) is modified with a weighting parameter λ that enables the relevance of the covariance term to be adjusted,

$$\tilde{\mathbf{G}} = \mathbb{E}[\mathbf{J}^T]\,\mathbb{E}[\mathbf{J}] + \lambda \boldsymbol{\Sigma}_J. \tag{4.20}$$

Setting λ to be small should find trajectories that prioritise staying in the desired mode, whereas selecting a large λ should find trajectories that prioritise avoiding regions of the dynamics with high *epistemic uncertainty*.

Extension to Sparse Variational Gaussian Processes

The model in Chapter 3 is built upon sparse GP approximations, so the Jacobian in Equation (4.14) must be extended for such approximations.

Sparse Gaussian Process Jacobian To obtain the distribution over the Jacobian in the sparse variational Gaussian process setting, we first condition on the inducing variables $h(\boldsymbol{\xi}) \in \mathbb{R}^{M\times 1}$,

$$\mathbf{J}_* \mid h(\boldsymbol{\xi}) \sim \mathcal{N}\left(\frac{\partial \mu}{\partial \mathbf{x}_*} + \partial \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}h(\boldsymbol{\xi}), \partial^2\mathbf{K}_{**} - \partial\mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\partial\mathbf{K}_{M*}\right). \quad (4.21)$$

where the inducing variables' density is from the prior in Equation (4.12), so the covariance matrices are given by,

$$\mathbf{K}_{MM} = k(\boldsymbol{\xi}, \boldsymbol{\xi}) \in \mathbb{R}^{M\times M} \quad (4.22)$$

$$\partial\mathbf{K}_{*M} = \frac{\partial k(\mathbf{x}_*, \boldsymbol{\xi})}{\partial \mathbf{x}_*} \in \mathbb{R}^{D\times M}. \quad (4.23)$$

The distribution over the Jacobian is then obtained by marginalising the inducing variables with respect to their variational density,

$$q(h(\boldsymbol{\xi})) = \mathcal{N}(h(\boldsymbol{\xi}) \mid \mathbf{m}, \mathbf{S}). \quad (4.24)$$

The distribution over the Jacobian is then obtained via a Gaussian convolution,

$$\mathbf{J}_* \sim \mathcal{N}\left(\underbrace{\frac{\partial \mu}{\partial \mathbf{x}_*} + \partial \mathbf{K}_{*M} \mathbf{K}_{MM}^{-1} \mathbf{m}}_{\boldsymbol{\mu}_\mathbf{J}}, \underbrace{\partial^2 \mathbf{K}_{**} - \partial \mathbf{K}_{*M} \mathbf{K}_{MM}^{-1} (\mathbf{K}_{MM} - \mathbf{S}) \mathbf{K}_{MM}^{-1} \partial \mathbf{K}_{M*}}_{\boldsymbol{\Sigma}_\mathbf{J}} \right). \tag{4.25}$$

4.2.2 Indirect Optimal Control via Latent Geodesics (IG)

This section presents a trajectory optimisation algorithm that exploits the fact that length minimising trajectories on the manifold endowed with the expected metric from Equation (4.19), encodes all of the goals. As shortest lengths on a manifold are known as geodesics, we refer to them as geodesic trajectories. The algorithm presented in this section exploits a classic result of Riemannian geometry, that geodesic trajectories are solutions to a 2^{nd} order ODE, known as the geodesic ODE f_G. As solutions to this ODE encode the necessary conditions for finding length minimising trajectories, the method presented in this section resembles an indirect optimal control method.

Geodesics Given the method for calculating lengths on Riemannian manifolds in Equation (4.9), the notion of a shortest trajectory, or geodesic trajectory, is defined as follows,

Definition 4.2.2 (Geodesic). *Given two points $\mathbf{x}_0, \mathbf{x}_f \in \mathcal{M} = h(\mathcal{X})$, a Geodesic is a length minimising trajectory (curve) $\bar{\mathbf{x}}_g$ connecting the points, such that,*

$$\bar{\mathbf{x}}_g = \arg\min_{\bar{\mathbf{x}}} \text{Length}(h(\bar{\mathbf{x}})) \tag{4.26a}$$

$$s.t. \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{4.26b}$$

$$\mathbf{x}(t_f) = \mathbf{x}_f. \tag{4.26c}$$

Geodesic ODE An important observation from Carmo, 1992, is that geodesics satisfy a continuous-time 2nd order ODE, given by,

$$\begin{aligned}\ddot{\mathbf{x}}(t) &= f_G(t, \mathbf{x}, \dot{\mathbf{x}}) \\ &= -\frac{1}{2}\mathbf{G}^{-1}(\mathbf{x}(t))\left[\frac{\partial \operatorname{vec}[\mathbf{G}(\mathbf{x}(t))]}{\partial \mathbf{x}(t)}\right]^T (\dot{\mathbf{x}}(t) \otimes \dot{\mathbf{x}}(t)),\end{aligned} \tag{4.27}$$

where $\operatorname{vec}[\mathbf{G}(\mathbf{x}(t))]$ stacks the columns of $\mathbf{G}(\mathbf{x}(t))$ and $\otimes$ denotes the Kronecker product. The implication of Equations (4.26) and (4.27), is that trajectories that are solutions to the 2nd order ODE in Equation (4.27), implicitly minimise their length on the manifold, i.e. the objective in Equation (4.9). Given this observation, computing geodesics involves finding a solution to Equation (4.27) with $\mathbf{x}(t_0) = \mathbf{x}_0$ and $\mathbf{x}(t_f) = \mathbf{x}_f$. This is a boundary value problem (BVP) with a smooth solution so it can be solved using any BVP solver, e.g. (multiple) shooting or collocation methods.

Implicit Trajectory Optimisation

Solving the 2nd order ODE in Equation (4.27) with the expected metric from Equation (4.20), is equivalent to solving our trajectory optimisation problem subject to the same boundary conditions. This resembles an indirect optimal control method as it is based on an observation that the necessary conditions for optimality are encoded via the geodesic ODE. However, it is worth noting that solutions to the geodesic ODE are not guaranteed to satisfy the dynamics constraints.

Collocation Since neither $\dot{\mathbf{x}}(t_0)$ nor $\dot{\mathbf{x}}(t_f)$ are known, Equation (4.27) cannot be solved with simple forward or backward integration. Instead, the problem is transcribed using collocation. Collocation methods are used to transcribe continuous-time trajectory optimisation problems into nonlinear programs, i.e. constrained parameter optimisations (Fahroo and Ross, 2000; Kelly, 2017). The expected metric in Equation (4.19) is substituted into Equation (4.27) and solved via collocation. This work implements a Hermite-Simpson collocation method. It parameterises the

state trajectory using cubic polynomials and the dynamics equations (the geodesic ODE in this case) are imposed as constraints at a set of collocation points. The trajectory $[t_0, t_f]$ is discretised into I intervals where the collocation points are the mid points of the discretisation intervals. The collocation states $\bar{\mathbf{z}} = \{\mathbf{z}_{i+\frac{1}{2}}\}_{i=0}^{I-1}$ are obtained by interpolating the polynomials. The derivative of the collocation states w.r.t. time $\{\dot{\mathbf{z}}_{i+\frac{1}{2}}, \ddot{\mathbf{z}}_{i+\frac{1}{2}}\}_{i=0}^{I-1}$ are obtained algebraically via the polynomials. The collocation constraints then enforce the second derivative of the collocation states interpolated by the polynomials $\{\ddot{\mathbf{z}}_{i+\frac{1}{2}}\}_{i=0}^{I-1}$, to equal the geodesic ODE f_G at the collocation points. This is achieved through the collocation defects,

$$\Delta\ddot{\mathbf{z}}_{i+\frac{1}{2}} = \ddot{\mathbf{z}}_{i+\frac{1}{2}} - f_G(t_{i+\frac{1}{2}}, \mathbf{z}_{i+\frac{1}{2}}, \dot{\mathbf{z}}_{i+\frac{1}{2}}) = 0 \quad \forall i \in \{0, \dots, I-1\}, \tag{4.28}$$

where $\ddot{\mathbf{z}}_{i+\frac{1}{2}}, \dot{\mathbf{z}}_{i+\frac{1}{2}}, \mathbf{z}_{i+\frac{1}{2}}$ are obtained by interpolating between i and $i+1$. Equation (4.28) defines a set of constraints that ensure trajectories are solutions to the geodesic ODE f_G. The nonlinear program that this method solves is given by,

$$\min_{\mathbf{z}_0 \dots \mathbf{z}_I, \dot{\mathbf{z}}_0 \dots \dot{\mathbf{z}}_I} \sum_{i=0}^{I-1} c(\mathbf{z}_i, \dot{\mathbf{z}}_i) \tag{4.29a}$$

$$\text{s.t.} \quad \Delta\ddot{\mathbf{z}}_{i+\frac{1}{2}} = 0 \quad \forall i \in \{0, \dots, I-1\} \tag{4.29b}$$

$$\mathbf{z}_0 = \mathbf{x}_0 \tag{4.29c}$$

$$\mathbf{z}_I = \mathbf{x}_f. \tag{4.29d}$$

Notice that no integrals need to be computed as all of the functions are algebraic operations. In practice, a quadratic cost function is used to regularise the state derivative $\dot{\mathbf{z}}$,

$$c(\mathbf{z}_i, \dot{\mathbf{z}}_i) = \dot{\mathbf{z}}_i^T \mathbf{R} \dot{\mathbf{z}}_i = \|\dot{\mathbf{z}}_i\|_{\mathbf{R}}, \tag{4.30}$$

where $\mathbf{R}$ is a user-defined, real symmetric positive definite weight matrix. It is solved using Sequential Least Squares Programming (SLSQP) in SciPy (Virtanen et al., 2020).

Latent variable controls This nonlinear program returns a collocation state trajectory $\bar{\mathbf{z}}$ which parameterises a continuous-time state trajectory (via the polynomials). However, it does not return the control trajectory. The control trajectory is recovered from the state trajectory by performing inference in the probabilistic dynamics model. In order to do this, the state trajectory is first discretised. In practice, using the collocation states as the discretised state trajectory worked well, i.e. $\bar{\mathbf{x}} = \{\mathbf{x}_t\}_{t=0}^{I} = \{\mathbf{z}_i\}_{i=0}^{I}$. The state difference outputs $\Delta\bar{\mathbf{x}} = \{\Delta\mathbf{x}_t\}_{t=1}^{T}$ are calculated from the state trajectory $\bar{\mathbf{x}}$. The control trajectory $\bar{\mathbf{u}}$ is then inferred from the state trajectory by extending the ELBO for the desired mode's SVGP expert with latent variable inputs. Following Hensman et al., 2013, the ELBO for a single SVGP expert is given by,

$$\log p(\Delta\bar{\mathbf{x}} \mid \bar{\mathbf{x}}, \bar{\mathbf{u}}) \geq \mathbb{E}_{q(f_k(\bar{\mathbf{x}},\bar{\mathbf{u}}))}[\log p(\Delta\bar{\mathbf{x}} \mid f_k(\bar{\mathbf{x}}, \bar{\mathbf{u}})) - \mathrm{KL}(q(f_k(\boldsymbol{\zeta}_k)) \mid p(f_k(\boldsymbol{\zeta}_k)))] \coloneqq \mathcal{L}_{\mathrm{SVGP}}, \tag{4.31}$$

where the variational posterior is given by $q(f_k(\bar{\mathbf{x}}, \bar{\mathbf{u}})) = \int p(f_k(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \mid f_k(\boldsymbol{\zeta}_k)) q(f_k(\boldsymbol{\zeta}_k)) \mathrm{d}f_k(\boldsymbol{\zeta}_k)$. The control inputs $\bar{\mathbf{u}}$ are recovered by treating them as latent variables and extending the lower bound to,

$$\log p(\Delta\bar{\mathbf{x}} \mid \bar{\mathbf{x}}) = \log \int p(\Delta\bar{\mathbf{x}} \mid \bar{\mathbf{x}}, \bar{\mathbf{u}}) p(\bar{\mathbf{u}}) \mathrm{d}\bar{\mathbf{u}} \tag{4.32}$$

$$\geq \mathbb{E}_{q(\bar{\mathbf{u}})}[\mathcal{L}_{\mathrm{SVGP}} + \log p(\bar{\mathbf{u}}) - \log q(\bar{\mathbf{u}})], \tag{4.33}$$

where each time step of the latent control trajectory is assumed to be normally distributed,

$$p(\bar{\mathbf{u}}) = \prod_{t=0}^{T-1} \mathcal{N}(\mathbf{u}_t \mid \mathbf{0}, \mathbf{I}), \tag{4.34}$$

and its variational posterior is given by,

$$q(\bar{\mathbf{u}}) = \prod_{t=0}^{T-1} \mathcal{N}(\mathbf{u}_t \mid \mathbf{m}_t, \mathbf{S}_t). \tag{4.35}$$

The posterior over the latent control trajectory $q(\bar{\mathbf{u}}) \approx p(\bar{\mathbf{u}} \mid \bar{\mathbf{x}}, \Delta\bar{\mathbf{x}})$ is obtained by finding the variational parameters $\{\mathbf{m}_t, \mathbf{S}_t\}_{t=0}^{T-1}$ that maximise the ELBO in Equation (4.33).

Although this method provides an elegant solution to finding trajectories that satisfy Goals 1, 2 and 3, it is not without its limitations. First of all, this approach does not necessarily find trajectories that satisfy the dynamics constraints, as it projects the problem onto the geodesic ODE.

Remark. *Dynamics constraints are not guaranteed to be satisfied.*

Secondly, it does not consider the full distribution over state-control trajectories. Without the inclusion of the full probabilistic dynamics model, it is impossible to consider the full distribution over state-control trajectories. Although propagating uncertainty through a single dynamics GP is straightforward, handling the collocation constraints is not. This is because the geodesic ODE will become a Stochastic Differential Equation (SDE).

Remark. *Ignores much of the stochasticity inherent in the problem.*

4.2.3 Direct Optimal Control via Riemannian Energy (DRE)

This section details a direct optimal control approach which embeds the mode remaining behaviour directly into the Stochastic Optimal Control (SOC) problem, via a geometric objective function. In contrast to the previous approach, this method:

1. enforces the dynamics constraints,
2. principally handles the uncertainty associated with the dynamics.

This approach is a shooting method that enforces the dynamics constraints through simulation, i.e. the state trajectory is enforced to match the integral of the dynamics with respect to time.

Similar to the previous approach, this method builds on the observation that length minimising trajectories on the Riemannian manifold $\mathcal{M}$, associated with the desired mode's gating function h_{k^*}, encodes the goals. Further to this, this method exploits the fact that length minimising trajectories on a Riemannian manifold $\mathcal{M}$, are also energy minimising trajectories (Carmo, 1992). As such, mode remaining behaviour can be encoded by solving,

$$\min_{\bar{\mathbf{u}}} J_\pi(\mathbf{x}_0) \tag{4.36a}$$

$$\text{s.t. } \mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \qquad \forall t \in \{0, \dots, T-1\} \tag{4.36b}$$

$$\mathbf{u}_t \in \mathcal{U} \qquad \forall t \in \{0, \dots, T-1\} \tag{4.36c}$$

$$\mathbf{x}_0 = \mathbf{x}_0 \tag{4.36d}$$

with an objective function that minimises the Riemannian energy,

$$J_\pi(\mathbf{x}) = \mathbb{E}\left[\underbrace{\text{Energy}(h(\bar{\mathbf{x}}))}_{\text{Riemannian energy}} + \underbrace{(\mathbf{x}_T - \mathbf{x}_f)^T \mathbf{H} (\mathbf{x}_T - \mathbf{x}_f)}_{\text{terminal cost}} + \sum_{t=0}^{T-1} \underbrace{\mathbf{u}_t^T \mathbf{R} \mathbf{u}_t}_{\text{control cost}} \mid \mathbf{x}_0 = \mathbf{x} \right], \tag{4.37}$$

where $\mathbf{H}$ and $\mathbf{R}$ are user-defined, real, symmetric, positive semi-definite and positive definite matrices respectively. The energy of a trajectory on a Riemannian manifold, endowed with the metric $\mathbf{G}$, is given by,

$$\text{Energy}(h(\bar{\mathbf{x}})) = \sum_{t=1}^{T} \Delta \mathbf{x}_t^T \mathbf{G}_{\mathbf{x}_t} \Delta \mathbf{x}_t, \tag{4.38}$$

where $\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$ is the state difference. The mode remaining behaviour and the terminal state boundary condition are encoded via the objective function.

Remark. *In contrast to the collocation solver in Section 4.2.2, the terminal state boundary condition is encoded via the cost function, instead of being enforced by the solver.*

This may seem like an easy optimisation problem, however, calculating the expected cost in Equation (4.37) is not straightforward. Given a starting state $\mathbf{x}_0$ and a control trajectory $\bar{\mathbf{u}}$, the expectation in Equation (4.37) is taken with respect to the joint state-metric distribution over a trajectory, $p(\bar{\mathbf{x}}, \bar{\mathbf{G}}, \mid \mathbf{x}_0, \bar{\mathbf{u}})$. Calculating this expectation is difficult as multi-step predictions in the MoSVGPE dynamics model cannot be calculated in closed form.

This work adopts a two-stage approximation to obtain a closed-form expression for the expected cost. First, multi-step dynamics predictions are approximated to obtain normally distributed states at each time step. Given normally distributed states, calculating the expected terminal and control cost terms in Equation (4.37) is straightforward. However, the expected Riemannian energy in Equation (4.37) has no closed-form expression, due to the dependence of metric $\mathbf{G}$ on the state. The second stage approximates the calculation of the expected Riemannian energy under normally distributed states.

Approximate Inference for Dynamics Predictions

Multi-step predictions in the MoSVGPE dynamics model have no closed-form solution because the state difference after the first time step is a Gaussian mixture, and propagating Gaussian mixtures through Gaussian processes has no closed-form solution. Further to this, constructing approximate closed-form solutions is difficult, due to the exponential growth in the number of Gaussian components.

Consider assuming each of the K dynamics modes to be independent. Recursively propagating the Gaussian components associated with the state, through all of the modes, over a trajectory of length T, would lead to the final state consisting of K^T Gaussian components.

This work sidesteps this issue and obtains closed-form multi-step predictions by enforcing that the controlled system remains in the desired dynamics mode. Multi-step predictions can then be calculated in closed-form by cascading single-step predictions using the desired dynamics GP, whose transition density is given by,

$$p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \alpha_t = k^*) = \mathcal{N}\big(f_{k^*}(\hat{\mathbf{x}}_t) \mid \mathbf{A}_{k^*}\mathbf{m}_{k^*}, k_{k^*}(\hat{\mathbf{x}}_t, \hat{\mathbf{x}}_t) + (\mathbf{S}_{k^*} - k_{k^*}(\boldsymbol{\zeta}_{k^*}, \boldsymbol{\zeta}_{k^*}))\mathbf{A}_{k^*}^T\big) \tag{4.39}$$

where $\mathbf{A}_{k^*} = k_{k^*}(\hat{\mathbf{x}}_t, \boldsymbol{\zeta}_{k^*})k_{k^*}(\boldsymbol{\zeta}_{k^*}, \boldsymbol{\zeta}_{k^*})^{-1}$ and $\hat{\mathbf{x}}_t = (\mathbf{x}_t, \mathbf{u}_t)$. Cascading single-step predictions requires recursively mapping uncertain state-control inputs through the desired mode's dynamics GP, i.e. recursively calculating the following integral,

$$p(\mathbf{x}_{t+1} \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \boldsymbol{\alpha}_{0:t} = k^*) = \int p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \alpha_t = k^*)p(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{u}_{0:t-1}, \boldsymbol{\alpha}_{0:t-1} = k^*)\mathrm{d}\mathbf{x}_t \tag{4.40}$$

with $p(\mathbf{x}_0) = \delta(\mathbf{x}_0)$. Approximate closed-form solutions exist for propagating normally distributed states and controls through GP models (Girard, 2004; Kuss, 2006; Quinonero-Candela et al., 2003). This work exploits the moment-matching approximation from Section 7.2.1 of Kuss, 2006.

$\delta-$ mode remaining chance constraints Enforcing the controlled system to remain in the desired dynamics mode simplifies calculating multi-step predictions and the expected cost in Equation (4.37). As the dynamics model is learned from observations, this work relaxes the requirement to ensuring that trajectories are

δ − mode remaining (Definition 4.1.1). The conditions to be δ − mode remaining can be enforced with chance constraints,

$$\Pr(\alpha_t = k^* \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \boldsymbol{\alpha}_{0:t-1} = k^*) \geq 1 - \delta \quad \forall t \in \{0, \dots, T\}. \tag{4.41}$$

These constraints enforce the system to remain in the desired dynamics mode with satisfaction probability $p_\alpha = 1 - \delta$, at each time step. As the MoSVGPE model assumes that the mode indicator variable α depends on the state via the gating function, this probability is calculated as follows,

$$\begin{aligned}\Pr(\alpha_t = k^* \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \boldsymbol{\alpha}_{0:t-1} = k^*) = \int & \underbrace{\Pr(\alpha_t = k^* \mid \mathbf{h}(\mathbf{x}_t))}_{\text{Bernoulli/softmax likelihood}} \\ & \int \underbrace{q(\mathbf{h}(\mathbf{x}_t) \mid \mathbf{x}_t)}_{\text{approx posterior}} \underbrace{p(\mathbf{x}_t \mid \mathbf{x}_0, \mathbf{u}_{0:t-1}, \boldsymbol{\alpha}_{0:t-1} = k^*)}_{\text{state dist}} \mathrm{d}\mathbf{x}_t \mathrm{d}\mathbf{h}(\mathbf{x}_t)\end{aligned} \tag{4.42}$$

where $q(\mathbf{h}(\mathbf{x}_t) \mid \mathbf{x}_t)$ is the approximate posterior over the gating functions from Equation (3.41).

Approximate Riemannian Energy

Given this approach for simulating the MoSVGPE dynamics model, the state at each time step is normally distributed. Unlike the terminal and control cost terms in Equation (4.37), the expected Riemannian energy,

$$\mathbb{E}_{\bar{\mathbf{x}}, \bar{\mathbf{J}}}[\text{Energy}(\bar{\mathbf{x}})] = \sum_{t=1}^{T} \mathbb{E}\left[\Delta \mathbf{x}_t^T \, \mathbb{E}_{\mathbf{J}_{\mathbf{x}_t} | \mathbf{x}_t}\left[\mathbf{J}_{\mathbf{x}_t} \mathbf{J}_{\mathbf{x}_t}^T\right] \Delta \mathbf{x}_t\right], \tag{4.43}$$

has no closed-form expression under normally distributed states. This is because the metric tensor $\mathbf{G}$ depends on the Jacobian, which depends on the state. However, it is possible to approximate the expected energy to obtain a closed-form expression.

The distribution over the Jacobian when the input location in normally distributed $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\Sigma}_{\mathbf{x}_t})$ can be calculated in closed-form when using the Squared Exponen-

tial kernel. However, this work simplifies the problem and calculates the Jacobian at the state mean of each time step along a trajectory, i.e. $\mathbf{J}_{\mathbf{x}_t} \approx \frac{\partial h(\boldsymbol{\mu}_{\mathbf{x}_t})}{\partial \boldsymbol{\mu}_{\mathbf{x}_t}}$. The distribution over the Jacobian given deterministic inputs can be calculated using Equation (4.14).

Approximating the Jacobian to be independent of the state enables the expected metric tensor to be calculated in closed-form with Equation (4.20). Given this approximation, the Riemannian energy retains a quadratic form, so the expectation with respect to $\Delta\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\Delta\mathbf{x}_t}, \boldsymbol{\Sigma}_{\Delta\mathbf{x}_t})$ can be calculated with,

$$\begin{aligned}\mathbb{E}[\text{Energy}(h(\bar{\mathbf{x}}))] &\approx \sum_{t=1}^{T} \mathbb{E}_{\Delta\mathbf{x}_t}\left[\Delta\mathbf{x}_t^T \, \mathbb{E}_{\mathbf{J}_{\mathbf{x}_t}}[\mathbf{G}_{\mathbf{x}_t}] \Delta\mathbf{x}_t\right] \\ &= \sum_{t=1}^{T} \boldsymbol{\mu}_{\Delta\mathbf{x}_t}^T (\boldsymbol{\mu}_{\mathbf{J}}\boldsymbol{\mu}_{\mathbf{J}}^T + \boldsymbol{\Sigma}_{\mathbf{J}})\boldsymbol{\mu}_{\Delta\mathbf{x}_t} + \text{tr}\big((\boldsymbol{\mu}_{\mathbf{J}}\boldsymbol{\mu}_{\mathbf{J}}^T + \boldsymbol{\Sigma}_{\mathbf{J}})\boldsymbol{\Sigma}_{\Delta\mathbf{x}_t}\big) \end{aligned} \tag{4.44}$$

The expected metric tensor encourages trajectories to 1) follow contours on the desired mode's gating function, encoding Goal 2, i.e. mode remaining behaviour, and to 2) avoid entering regions where it is uncertain which mode governs the dynamics, i.e. Goal 3.2. The expectation over the state difference then encourages trajectories to remain in regions of the desired dynamics mode with low uncertainty, i.e. Goal 3.1.

Given this approximation for the expected Riemannian energy, the expected cost in Equation (4.36) can be calculated in closed-form with,

$$J(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \underbrace{(\boldsymbol{\mu}_{\mathbf{x}_T} - \mathbf{x}_f)^T \mathbf{H} (\boldsymbol{\mu}_{\mathbf{x}_T} - \mathbf{x}_f) + \mathrm{tr}(\mathbf{H}\boldsymbol{\Sigma}_{\mathbf{x}_T})}_{\text{expected terminal cost}} \tag{4.45}$$

$$+ \underbrace{\sum_{t=1}^{T} \boldsymbol{\mu}_{\Delta\mathbf{x}_t}^T (\boldsymbol{\mu}_\mathbf{J} \boldsymbol{\mu}_\mathbf{J}^T + \boldsymbol{\Sigma}_\mathbf{J}) \boldsymbol{\mu}_{\Delta\mathbf{x}_t} + \mathrm{tr}\big((\boldsymbol{\mu}_\mathbf{J} \boldsymbol{\mu}_\mathbf{J}^T + \boldsymbol{\Sigma}_\mathbf{J}) \boldsymbol{\Sigma}_{\Delta\mathbf{x}_t}\big)}_{\text{expected Riemannian energy}} \tag{4.46}$$

$$+ \underbrace{\sum_{t=0}^{T-1} \boldsymbol{\mu}_{\mathbf{u}_t}^T \mathbf{R} \boldsymbol{\mu}_{\mathbf{u}_t} + \mathrm{tr}(\mathbf{R}\boldsymbol{\Sigma}_{\mathbf{u}_t})}_{\text{expected control cost}} . \tag{4.47}$$

This work then approximately solves the problem in Equation (4.1) by solving,

$$\min_{\bar{\mathbf{u}}} \quad J(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \tag{4.48a}$$

$$\text{s.t.} \quad \text{Equations (4.40) and (4.41)} \tag{4.48b}$$

using SLSQP in SciPy (Virtanen et al., 2020). This method obtains closed-form expressions for the expected cost in Equation (4.36) by constraining the system to be $\delta-$mode remaining (Definition 4.1.1).

Practical Implementation

An alternative approach to obtain mode remaining behaviour is to optimise subject to the chance constraints in Equation (4.41) alone, i.e. without the Riemannian energy cost term. However, this constrained optimisation is often not able to converge in practice. Experiments and intuition indicate that the geometry of the gating functions provides a much better optimisation landscape. This is because the gating functions vary gradually over the state domain, whilst the mixing probability changes abruptly at the boundaries between dynamics modes.

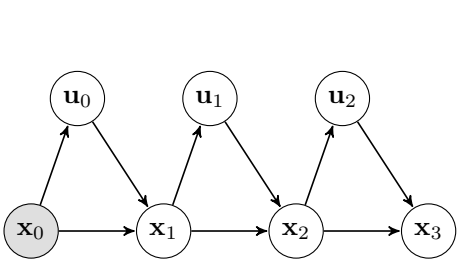
Therefore in practice, the optimisation in Equation (4.48) is performed unconstrained, i.e. without enforcing the chance constraints at every iteration. Instead, the chance constraints are used to validate trajectories found by the unconstrained optimiser, before deploying them in the environment. In most experiments, this strategy was far superior than constraining the optimisation at every iteration.

4.3 Mode Remaining Control as Probabilistic Inference

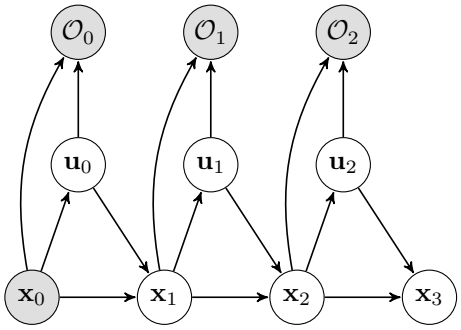
This section presents an alternative approach to finding mode remaining trajectories, named Mode Remaining Control as Inference (MRCaI). In contrast to the previous section that encoded mode remaining behaviour via the latent geometry of the MoSVGPE's gating network, this section unleashes the power of the probability mass function over the expert indicator variable. As all MoGPE methods have a probability mass function over the expert indicator variable, the method presented in this chapter is applicable in a wider range of MoGPE dynamics models. Section 4.3.1 recaps the necessary background and related work and Section 4.3.2 then details the trajectory optimisation algorithm.

4.3.1 Background and Related Work

This section first recaps the Control as Inference (CaI) framework. To formulate optimal control as probabilistic inference it is first embedded into a graphical model (see Figure 4.2a). The joint probability model (over a trajectory) is augmented with an additional variable to encode the notion of cost (or reward) over the trajectory (see Figure 4.2b). The new variable is a Bernoulli random variable $\mathcal{O}_t \in \{0, 1\}$, that indicates if time step t is *optimal* $\mathcal{O}_t = 1$, or not *optimal* $\mathcal{O}_t = 0$. The likelihood dis-



(a) Graphical model with states and controls.



(b) Graphical model with optimality variable.

Figure 4.2: *Graphical models of control formulated as inference.*

tribution can be formulated by mapping the negative cost through a monotonically increasing function g, giving the likelihood,

$$\Pr(\mathcal{O}_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t) \coloneqq g(-c(\mathbf{x}_t, \mathbf{u}_t)). \tag{4.49}$$

A common (and convenient) approach is to formulate the likelihood using an exponential transform of the cost. This results in a Boltzmann distribution where the inverse temperature, γ, is used to scale the cost,

$$\Pr(\mathcal{O}_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t) \propto \exp(-\gamma c(\mathbf{x}_t, \mathbf{u}_t)). \tag{4.50}$$

The resulting negative log-likelihood, for a single state-control trajectory $\bar{\mathbf{x}}$, $\bar{\mathbf{u}}$, is an affine transformation of the cost,

$$-\log \Pr(\bar{\boldsymbol{\mathcal{O}}} \mid \bar{\mathbf{x}}, \bar{\mathbf{u}}) = \gamma c(\bar{\mathbf{x}}, \bar{\mathbf{u}}), \tag{4.51}$$

which preserves convexity. The set of optimal Bernoulli variables over a trajectory is denoted $\bar{\boldsymbol{\mathcal{O}}} = \{\mathcal{O}_t = 1\}_{t=0}^{T}$. When the inverse temperature parameter is set to $\gamma = 1$ the maximum likelihood trajectory coincides with classical optimal control (Toussaint, 2009).

Inference of Sequential Latent Variables

The joint probability for an optimal trajectory (i.e. for $\mathcal{O}_t = 1$ for all $t \in \{0, \dots, T\}$), can be factorised using its Markovian structure,

$$p(\bar{\boldsymbol{\mathcal{O}}}, \bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0) = \left[\prod_{t=0}^{T} \underbrace{\Pr(\mathcal{O}_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t)}_{\text{cost}}\right] \left[\prod_{t=0}^{T-1} \underbrace{p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)}_{\text{dynamics}} \underbrace{\pi(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{controller}}\right], \tag{4.52}$$

where $\pi(\mathbf{u}_t \mid \mathbf{x}_t)$ denotes the controller or policy. Toussaint, 2009 highlights that although the maximum likelihood trajectory coincides with the classical optimal trajectory, taking expectations over trajectories i.e. calculating $\log p(\bar{\boldsymbol{\mathcal{O}}} \mid \mathbf{x}_0)$, is not equivalent to expected cost minimisation. Rawlik et al., 2013 extend the concepts from Toussaint, 2009 to show the general relation to classical SOC. For a given policy π, they introduce the posterior distribution over state-control trajectories as,

$$p_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \bar{\boldsymbol{\mathcal{O}}}, \mathbf{x}_0) = Z^{-1} \left[\prod_{t=0}^{T} \exp(-\gamma c(\mathbf{x}_t, \mathbf{u}_t))\right] \left[\prod_{t=0}^{T-1} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \pi(\mathbf{u}_t \mid \mathbf{x}_t)\right], \tag{4.53}$$

where $Z = p(\bar{\boldsymbol{\mathcal{O}}} \mid \mathbf{x}_0)$. This distribution $p_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \bar{\boldsymbol{\mathcal{O}}}, \mathbf{x}_0)$ is conditioned on the optimality variable but generated by a potentially uniform policy π.

They then distinguish between a prior policy π_0 and an unknown control policy π. The prior distribution over state-control trajectories under the control policy π, is given by,

$$q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \prod_{t=0}^{T-1} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \pi(\mathbf{u}_t \mid \mathbf{x}_t). \tag{4.54}$$

Intuitively $q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, is thought of as the controlled process, which is not conditioned on optimality and $p_{\pi_0}(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \bar{\boldsymbol{\mathcal{O}}}, \mathbf{x}_0)$ as the posterior process, conditioned on optimality but generated by a potentially uniform policy, π_0. The dual problem is then

to find the control policy, π, where the controlled process, $q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, matches the posterior process, $p_{\pi_0}(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \bar{\boldsymbol{\mathcal{O}}}, \mathbf{x}_0)$. Given π_0 is an arbitrary stochastic policy and $\mathbb{D}$ is the set of deterministic policies, the problem,

$$\begin{aligned}
\pi_* &= \arg\min_{\pi \in \mathbb{D}} \mathrm{KL}(q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \,||\, p_{\pi_0}(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \bar{\boldsymbol{\mathcal{O}}}, \mathbf{x}_0)) \\
&= \arg\min_{\pi \in \mathbb{D}} Z + \gamma \, \mathbb{E}_{q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})}[c(\bar{\mathbf{x}}, \bar{\mathbf{u}})] + \mathbb{E}_{q_\pi(\bar{\mathbf{x}})}[\mathrm{KL}(\pi \,||\, \pi_0)] \\
&= \arg\min_{\pi \in \mathbb{D}} \underbrace{Z}_{\text{constant}} + \underbrace{\gamma \, \mathbb{E}_{q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})}[c(\bar{\mathbf{x}}, \bar{\mathbf{u}})]}_{\text{expected costs}} - \mathbb{E}_{q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})}[\log \pi_0(\bar{\mathbf{u}} \mid \bar{\mathbf{x}})] - \underbrace{\mathbb{E}_{q_\pi(\bar{\mathbf{x}})}[H[\pi]]}_{\text{max entropy term}},
\end{aligned} \tag{4.55}$$

is equivalent to the SOC problem, with a modified integral cost,

$$\hat{c}(\mathbf{x}_t, \mathbf{u}_t) = c(\mathbf{x}_t, \mathbf{u}_t) - \frac{1}{\gamma} \log \pi_0(\mathbf{u}_t \mid \mathbf{x}_t). \tag{4.56}$$

The problem in Equation (4.55) finds trajectories that balance minimising expected costs $\mathbb{E}_{q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})}[c(\bar{\mathbf{x}}, \bar{\mathbf{u}})]$ and selecting a policy π that is similar to the prior policy π_0. If the prior policy π_0 is assumed to be uniform, then $\mathbb{E}_{q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})}[\log \pi_0(\bar{\mathbf{u}} \mid \bar{\mathbf{x}})]$ becomes constant and the optimised policy, π_*, is a balance of minimising expected costs and maximising the policy's entropy,

$$\pi_* = \arg\min_{\pi \in \mathbb{D}} \underbrace{\gamma \, \mathbb{E}_{q_\pi(\bar{\mathbf{x}}, \bar{\mathbf{u}})}[c(\bar{\mathbf{x}}, \bar{\mathbf{u}})]}_{\text{expected costs}} - \underbrace{\mathbb{E}_{q_\pi(\bar{\mathbf{x}})}[H[\pi]]}_{\text{max entropy term}}. \tag{4.57}$$

Maximum entropy regularisation Formulating trajectory optimisation in this way encodes the maximum causal entropy principle, which is often used to achieve robustness, in particular for inverse optimal control (Ziebart, 2010).

Other approaches There are multiple approaches to performing inference in this graphical model. Trading accuracy for computational complexity is often required for real-time control. In this case, one approach is to approximate the dynamics with linear or quadratic approximations, as is done in iLQR/iLQG and GP respec-

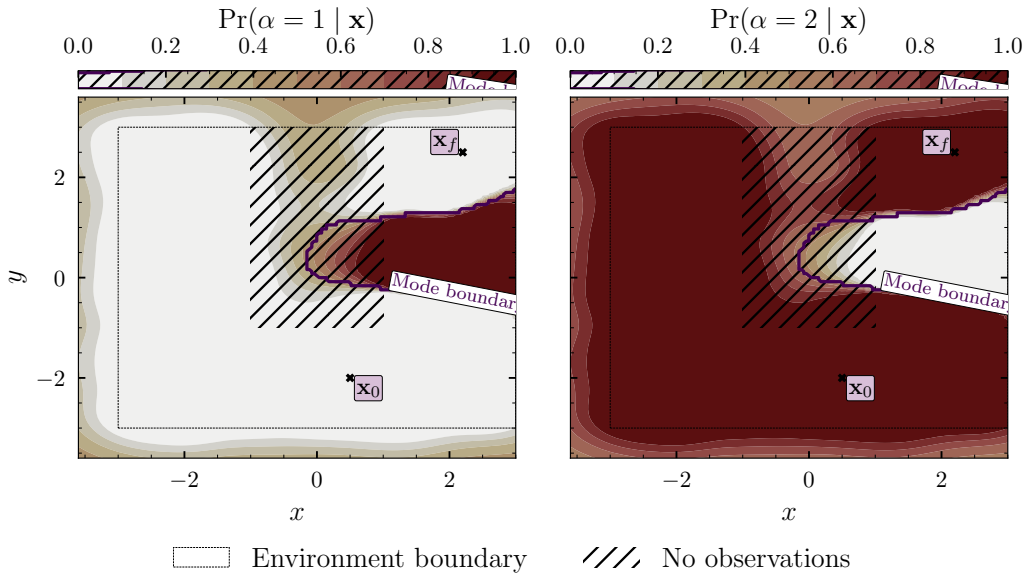


Figure 4.3: *Visualisation of the probability mass function over the expert indicator variable after training MoSVGPE on the state transition data set from the simulated version of the 2D quadcopter environment in the illustrative example from Section 1.1. The start state* $\mathbf{x}_0$ *and target state* $\mathbf{x}_f$ *are overlayed along with the mode boundary (purple line) and the subset of the environment which has not been observed (hashed box).*

tively. Given linear dynamics, the full graphical model in Equation (4.52) can be computed using approximate Gaussian message passing, for which effective methods exist (Loeliger et al., 2007). The inference problem can then be solved using the expectation maximisation algorithm for dynamical system estimation (Ghahramani and Roweis, 1999; Schön et al., 2011; Shumway and Stoffer, 1982), with input estimation (Watson et al., 2021).

4.3.2 Mode Remaining Control as Inference

This section details how the Control as Inference (CaI) framework can be extended to multimodal dynamical systems and used to encode mode remaining behaviour. Figure 4.3 shows the probability mass function over the expert indicator variable after training MoSVGPE on the historical data set of state transitions from the quadcopter navigation problem in Section 1.1. Intuitively, the goal is to find trajec-

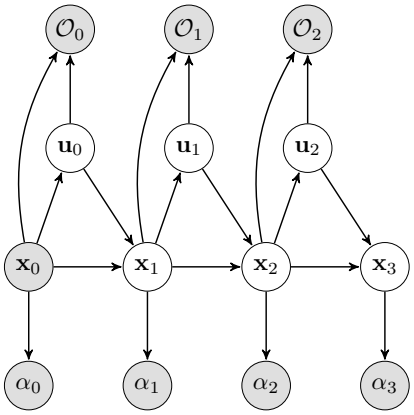


Figure 4.4: *Graphical model with optimality and mode indicator variables.*

tories that remain in regions of the dynamics with a high probability of remaining in the desired dynamics mode.

In order to find trajectories that remain in the desired dynamics mode, this work further augments the graphical model in Figure 4.2 with the mode indicator variable $\alpha \in \mathcal{A}$ from Equation (3.1). The resulting graphical model is shown in Figure 4.4, where the evidence is that $\mathcal{O}_t = 1$ and $\alpha_t = k^*$ for all $t \in \{0, \dots, T\}$. The joint probability model is then given by,

$$p(\bar{\boldsymbol{\mathcal{O}}}, \bar{\boldsymbol{\alpha}}, \bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0) = \Bigg[\prod_{t=0}^{T} \underbrace{\Pr(\mathcal{O}_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t)}_{\text{cost}} \underbrace{\Pr(\alpha_t = k^* \mid \mathbf{x}_t)}_{\text{mode remaining term}} \Bigg] \Bigg[\prod_{t=0}^{T-1} \underbrace{p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \alpha_t = k)}_{\text{dynamics}} \underbrace{\pi(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{controller}} \Bigg], \tag{4.58}$$

where $\bar{\boldsymbol{\alpha}} = \{\alpha_t = k^*\}_{t=0}^{T}$ denotes every time step of a trajectory belonging to the desired dynamics mode k^*. Equation (4.58) says that the probability of observing a trajectory is given by taking the product of its probability of occurring according to the dynamics, with the exponential of the negative cost and the probability of remaining in the desired dynamics mode. Given deterministic dynamics, the trajectory with the highest probability will be that with the lowest cost and highest probability of remaining in the desired dynamics mode.

This work draws on the connection between KL-divergence control (Rawlik et al., 2013) and structured variational inference. Whilst the derivation shown here differs from Rawlik et al., 2013, the underlying framework and objective are the same.

In variational inference, the goal is to approximate a distribution $p(\mathbf{y})$ with another, simpler distribution $q(\mathbf{y})$. Typically this distribution $q(\mathbf{y})$ is selected to be a product of conditional distributions connected in a chain or tree, which lends itself to tractable inference. In this work, the goal is to approximate the intractable distribution over optimal trajectories,

$$p(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\mathcal{O}}}, \bar{\boldsymbol{\alpha}}) = Z^{-1} \left[\prod_{t=0}^{T} \underbrace{\exp(-\gamma c(\mathbf{x}_t, \mathbf{u}_t))}_{\text{cost}} \underbrace{\Pr(\alpha_t = k^* \mid \mathbf{x}_t)}_{\text{mode remaining term}} \right] \left[\prod_{t=0}^{T-1} \underbrace{p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \alpha_t = k)}_{\text{dynamics}} \underbrace{\pi(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{controller}} \right] \tag{4.59}$$

with the variational distribution $q(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}})$. Note that $Z = p(\bar{\boldsymbol{\mathcal{O}}}, \bar{\boldsymbol{\alpha}} \mid \mathbf{x}_0)$. In this work the variational distribution over the controller is assumed independent of the state. That is, instead of parameterising the controller with state feedback, it is parameterised as a set of open-loop controls $\bar{\mathbf{u}} = \{\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}\}$ that are treated as random variables with density q($\bar{\mathbf{u}}$). Calculating $q(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}})$ for a set of controls $q(\bar{\mathbf{u}}) = \prod_{t=0}^{T-1} q(\mathbf{u}_t)$, requires simulating the trajectory in the learned, single-step dynamics model, i.e. making long-term predictions.

Approximate Inference for Dynamics Predictions

As this method is using a learned representation of the transition dynamics, it suffices to assume that the dynamics are given by the desired mode's learned dynamics. Constructing approximate closed-form solutions based on the model in Chapter 3 is difficult, due to the exponential growth in the number of Gaussian components. Similar to the approach in Section 4.2.3, this method obtains multi-step predictions by cascading single-step predictions through the desired mode's dynamics GP.

However, this approach extends the predictions to handle normally distributed controls $\mathbf{u}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_t}, \boldsymbol{\Sigma}_{\mathbf{u}_t})$. Multi-step predictions are then obtained by recursively calculating the following integral,

$$\underbrace{q(\mathbf{x}_{t+1} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t})}_{\text{next multi-step state dist}} = \int_{\mathcal{U}} \int_{\mathcal{X}} \underbrace{p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \alpha_t = k)}_{\text{single-step dynamics}} \underbrace{q(\mathbf{x}_t \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t-1})}_{\text{multi-step state dist}} \underbrace{q(\mathbf{u}_t)}_{\text{control}} \mathrm{d}\mathbf{x}_t \mathrm{d}\mathbf{u}_t, \tag{4.60}$$

using the moment matching approximation from (Kuss, 2006). Note that $\bar{\boldsymbol{\alpha}}_{0:t}$ denotes the first t elements of $\bar{\boldsymbol{\alpha}}$, i.e. $\bar{\boldsymbol{\alpha}}_{0:t} = \{\alpha_i = k^*\}_{i=0}^{t}$. Given this method for making multi-step predictions, the variational distribution over state-control trajectories is given by,

$$q(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}) = \prod_{t=0}^{T-1} q(\mathbf{x}_{t+1} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t}) q(\mathbf{u}_t), \tag{4.61}$$

with $q(\mathbf{x}_0) = \delta(\mathbf{x}_0)$. Note that the state and control at each time step are normally distributed.

Variational Inference for Sequential Latent Variables

Variational inference seeks to optimise $q(\bar{\mathbf{u}})$ w.r.t. the Evidence Lower Bound (ELBO). In this setup, the evidence is that $\mathcal{O}_t = 1$ and $\alpha_t = k^*$ for all $t \in \{0, \ldots, T\}$. Given this, the ELBO is given by,

$$
\begin{aligned}
\log p(\bar{\boldsymbol{\mathcal{O}}}, \bar{\boldsymbol{\alpha}} \mid \mathbf{x}_0) &= \log \mathbb{E}_{q(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}})} \left[\frac{p(\bar{\boldsymbol{\mathcal{O}}}, \bar{\boldsymbol{\alpha}}, \bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0)}{q(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}})} \right] \\
&= \log \mathbb{E}_{q(\bar{\mathbf{x}}, \bar{\mathbf{u}} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}})} \left[\frac{\prod_{t=0}^{T} [\Pr(\mathcal{O}_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t) \Pr(\alpha_t = k^* \mid \mathbf{x}_t)] \prod_{t=0}^{T-1} \pi(\mathbf{u}_t \mid \mathbf{x}_t)}{\prod_{t=0}^{T-1} q(\mathbf{u}_t)} \right] \\
&\geq - \sum_{t=0}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t-1}) q(\mathbf{u}_t)} [c(\mathbf{x}_t, \mathbf{u}_t)]}_{\text{expected cost}} + \sum_{t=0}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t-1})} [\log \Pr(\alpha_t = k^* \mid \mathbf{x}_t)]}_{\text{mode remaining term}} \\
&\quad - \sum_{t=0}^{T-1} \mathrm{KL}(q(\mathbf{u}_t) \mid\mid \pi(\mathbf{u}_t \mid \mathbf{x}_t)) \coloneqq \mathcal{L}_{\text{mode}},
\end{aligned} \tag{4.62}
$$

where the inequality is obtained via Jensen's inequality. Note the cancellation in the second line where we assume $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \alpha_t = k) = q(\mathbf{x}_{t+1} \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t})$. Assuming a uniform prior policy $\pi(\mathbf{u}_t \mid \mathbf{x}_t)$ leads to the KL term reducing to an entropy term and a constant,

$$
\begin{aligned}
\mathcal{L}_{\text{mode}} = &- \sum_{t=0}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t-1}) q(\mathbf{u}_t)} [c(\mathbf{x}_t, \mathbf{u}_t)]}_{\text{expected cost}} + \sum_{t=0}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t-1})} [\log \Pr(\alpha_t = k^* \mid \mathbf{x}_t)]}_{\text{mode remaining term}} \\
&+ \sum_{t=0}^{T-1} \underbrace{\cancel{\mathbb{E}_{q(\mathbf{x}_t \mid \mathbf{x}_0, \bar{\boldsymbol{\alpha}}_{0:t-1}) q(\mathbf{u}_t)} [\log \pi(\mathbf{u}_t \mid \mathbf{x}_t)]}}_{\text{constant}} + \sum_{t=0}^{T-1} \underbrace{H[\mathbf{u}_t]}_{\text{entropy}}.
\end{aligned} \tag{4.63}
$$

The ELBO in Equation (4.63) resembles the KL control objective in Equation (4.57) but with an extra term encoding the mode remaining behaviour. In practice, maximum entropy control is achieved by parameterising the control at each time step to be normally distributed $q(\mathbf{u}_t) = \mathcal{N}(\mathbf{u}_t \mid \boldsymbol{\mu}_{\mathbf{u}_t}, \boldsymbol{\Sigma}_{\mathbf{u}_t})$. The control dimensions are assumed independent so $\boldsymbol{\Sigma}_{\mathbf{u}_t}$ becomes diagonal. The maximum entropy behaviour

can be omitted by using deterministic controls, i.e. parameterising them to follow a Dirac delta distribution $q(\mathbf{u}_t) = \delta(\mathbf{u}_t)$. The control problem is then given by,

$$\max_{\bar{\mathbf{u}}} -\sum_{t=0}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0,\bar{\boldsymbol{\alpha}}_{0:t-1})q(\mathbf{u}_t)}[c(\mathbf{x}_t,\mathbf{u}_t)]}_{\text{expected cost}} + \sum_{t=0}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0,\bar{\boldsymbol{\alpha}}_{0:t-1})}[\log \Pr(\alpha_t = k^* \mid \mathbf{x}_t)]}_{\text{mode remaining term}} + \sum_{t=0}^{T-1} \underbrace{H[\mathbf{u}_t]}_{\text{entropy}} \tag{4.64a}$$

$$\text{s.t. Equation (4.60)} \tag{4.64b}$$

which encodes mode remaining behaviour alongside maximum entropy control. The variational parameters $\boldsymbol{\mu}_{\mathbf{u}_t}$ (and $\boldsymbol{\Sigma}_{\mathbf{u}_t}$) are found by maximising the ELBO using gradient-based optimisation. At each iteration, the ELBO is calculated by rolling out the control distribution in the desired mode's GP dynamics model using Equation (4.60). The resulting state-control trajectory is then used to calculate the ELBO. The cost function is given by,

$$c(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \underbrace{(\mathbf{x}_T - \mathbf{x}_f)^T \mathbf{H} (\mathbf{x}_T - \mathbf{x}_f)}_{\text{terminal cost}} + \sum_{t=0}^{T-1} \underbrace{\mathbf{u}_t^T \mathbf{R} \mathbf{u}_t}_{\text{control cost}} , \tag{4.65}$$

where $\mathbf{H}$ and $\mathbf{R}$ are user-defined, real, symmetric, positive semi-definite and positive definite matrices respectively. This cost function encodes the terminal state boundary condition in Equation (4.1) via a quadratic cost that minimises the deviation of the final state $\mathbf{x}_T$ from the target state $\mathbf{x}_f$. Importantly, the expected value of this cost under normally distributed states and controls can be calculated in closed-form with,

$$\mathbb{E}_{\bar{\mathbf{x}},\bar{\mathbf{u}}}[c(\bar{\mathbf{x}}, \bar{\mathbf{u}})] = (\boldsymbol{\mu}_{\mathbf{x}_T} - \mathbf{x}_f)^T \mathbf{H} (\boldsymbol{\mu}_{\mathbf{x}_T} - \mathbf{x}_f) + \operatorname{tr}(\mathbf{H}\boldsymbol{\Sigma}_{\mathbf{x}_T}) + \sum_{t=1}^{T-1} \boldsymbol{\mu}_{\mathbf{u}_t}^T \mathbf{R} \boldsymbol{\mu}_{\mathbf{u}_t}. + \sum_{t=1}^{T-1} \operatorname{tr}(\mathbf{R}\boldsymbol{\Sigma}_{\mathbf{u}_t}) \tag{4.66}$$

4.4 Conclusion

This chapter has presented three *mode remaining* trajectory optimisation algorithms. The first two have shown how the geometry of the MoSVGPE gating network infers valuable information regarding how a multimodal dynamical system switches between its underlying dynamics modes. Moreover, they have shown how this latent geometry can be leveraged to encode *mode remaining* behaviour into two different control strategies. Both of these control strategies introduce a user tunable parameter λ that can be tuned to either prioritise remaining in the desired dynamics mode or avoiding regions of high *epistemic uncertainty*. However, Chapter 5 will show that setting the λ parameter is not straightforward in practice. The third method presented in this chapter has shown how the probability mass function over the MoGPE's expert indicator variable can be used to encode mode remaining behaviour for trajectory optimisation. In particular, it has shown how the MRCaI framework (Kappen et al., 2013; Toussaint, 2009; Toussaint and Storkey, 2006) can be extended to multimodal dynamical systems and how mode remaining behaviour can be encoded by conditioning on the mode indicator variable.

In Chapter 5 the methods presented in this chapter are evaluated and compared using the quadcopter navigation problem from the illustrative example in Section 1.1. It turns out that in practice, the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3, perform significantly better than the Indirect Optimal Control via Latent Geodesics (IG) method.

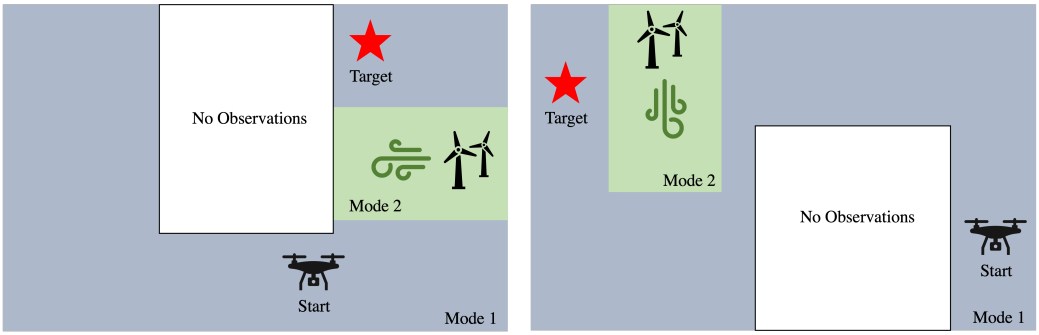
5 Quadcopter Experiments - Mode Remaining Trajectory Optimisation

This chapter evaluates the three *mode remaining* trajectory optimisation algorithms:

1. Indirect Optimal Control via Latent Geodesics (IG) from Section 4.2.2,
2. Direct Optimal Control via Riemannian Energy (DRE) from Section 4.2.3,
3. Mode Remaining Control as Inference (MRCaI) from Section 4.3.2,

in a simulated version of the illustrative example from Section 1.1. That is, flying a velocity controlled quadcopter from an initial state $\mathbf{x}_0$, to a target state $\mathbf{x}_f$, whilst avoiding the turbulent dynamics mode. The methods are further evaluated on a second simulated velocity controlled quadcopter navigation problem. See Figure 5.1 for schematics of the two environments. The turbulent dynamics modes (green) are subject to higher drift due to the wind field created by the fan. They are also subject to higher diffusion (process noise) resulting from the turbulence induced by the fan. Although the exact turbulent dynamics are *unknown*, they are believed to be difficult to control. This is due to the high process noise which may lead to catastrophic failure. Therefore, it is desirable to find trajectories that avoid entering this turbulent dynamics mode.

The collocation solver from the Indirect Optimal Control via Latent Geodesics (IG) method in Section 4.2 is tested on a real-world quadcopter state transition data set. This data set was collected with constant controls so the controls could not



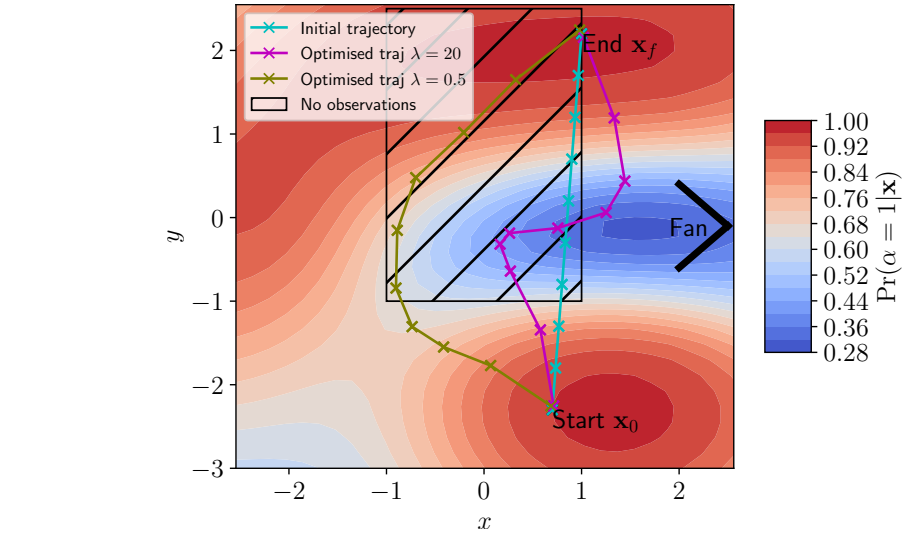
(a) Environment 1 *(b) Environment 2*

Figure 5.1: *Visualisation of two quadcopter navigation problems in environments with different spatially varying modes. It shows the desired dynamics mode in blue (Mode 1) and the turbulent dynamics mode induced by a fan in green (Mode 2). The white box indicates a region of the environment which was not observed. The goal is to navigate from the start state $\mathbf{x}_0$, to the target state $\mathbf{x}_f$, whilst remaining in the desired dynamics mode (Mode 1).*

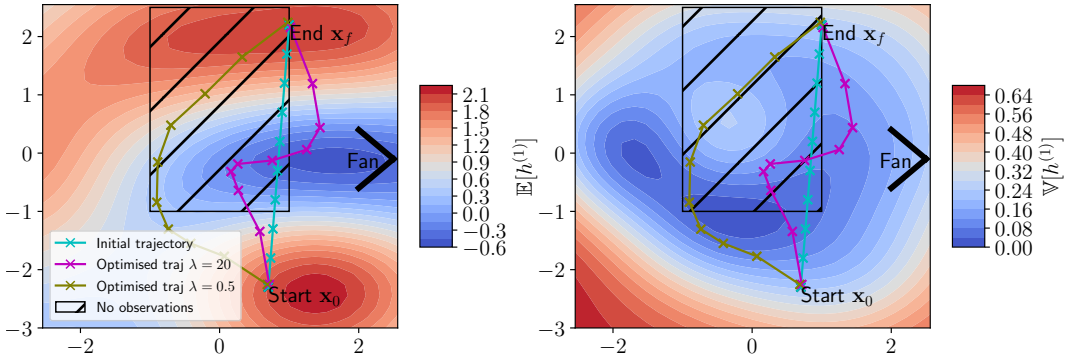
be recovered from the full MoSVGPE dynamics model, nor could the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2 or the Mode Remaining Control as Inference (MRCaI) method from Section 4.3 be tested. Nevertheless, the results are a small step towards validating the method's applicability to real-world systems. All three control methods are tested in the two simulated environments so that they can be compared. To aid comparison with the real-world experiments, the layout of the first simulated environment (Environment 1) was kept consistent with the real-world experiments.

5.1 Real-World Quadcopter Experiments

The Indirect Optimal Control via Latent Geodesics (IG) method presented in Section 4.2.2 was evaluated using data from the real-world quadcopter navigation problem detailed in Section 3.5.3. However, a different subset of the environment was not observed and the model was trained using an old variational inference scheme, not the one presented in Chapter 3. Figure 5.1a shows the environment and details the quadcopter navigation problem. The controls were kept constant during data



(a) Desired mode's mixing probability.



(b) GP posterior mean (left) and variance (right) over the desired mode's gating function.

Figure 5.2: ***Indirect Optimal Control via Latent Geodesics (IG)*** *trajectory optimisation results after solving the nonlinear program in Equation (4.29) using the desired mode's ($\alpha = 1$) gating function from the MoSVGPE (with $K = 2$ experts) after training on the real-world velocity controlled quadcopter data set. The initial (cyan) and optimised trajectories' – for two settings of λ – are overlayed on the desired mode's (a) mixing probability and (b) gating function GP posterior.*

collection, reducing the dynamics to $\Delta\mathbf{x}_{t+1} = f(\mathbf{x}_t; \mathbf{u}_t = \mathbf{u}_*)$. See Section 3.5.3 for more details on data collection and processing.

5.1.1 Model Learning

The model from Chapter 3 was instantiated with $K = 2$ experts and trained on the data collected from the velocity controlled quadcopter experiment. Each mode's

dynamics GP used a Squared Exponential kernel with Automatic Relevance Determination (ARD) and a constant mean function. The gating network used a single gating function with a Bernoulli likelihood. Its GP prior utilised a Squared Exponential kernel with ARD and a zero mean function.

Figure 5.2 shows the gating network posterior where the model has learned two dynamics modes, characterised by the drift and process noise induced by the fan. Mode 1 (red) represents the operable dynamics mode whilst Mode 2 (blue) represents the inoperable turbulent dynamics mode. This is illustrated in Figure 5.2a which shows the probability that the desired mode ($\alpha = 1$) governs the dynamics over the domain. Figure 5.2b shows the GP posterior mean (left) and variance (right) of the gating function h_1 associated with the desired dynamics mode. The mean is high where the model believes the desired mode is responsible for predicting, low where it believes another mode is responsible and zero where it is uncertain. The variance (right) has also clearly captured information regarding the *epistemic uncertainty*, i.e. where the model is uncertain which mode governs the dynamics.

5.1.2 Trajectory Optimisation using Indirect Optimal Control via Latent Geodesics

The initial (cyan) trajectory in Figure 5.2 was initialised as a straight line with 10 collocation points, indicated by the crosses. The collocation solver guarantees that trajectories end at the target state. However, trajectories are not guaranteed to remain in the desired dynamics mode, nor are they guaranteed to satisfy the system dynamics. Table 5.1 compares the initial trajectory to the results obtained with two settings of the λ parameter from Equation (4.20), which determines the relevance of the covariance term in the expected Riemannian metric.

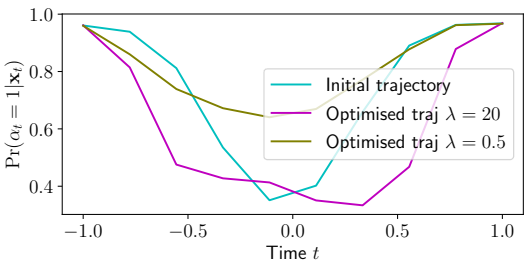
The higher probability of remaining in the desired dynamics mode indicates that the lower setting of $\lambda = 0.5$ exhibits more mode remaining behaviour. This can be seen visually in Figure 5.2a, where the trajectory with $\lambda = 0.5$ remains in regions of the model with high probability. The right-hand plot in Figure 5.2b shows that

Table 5.1: ***Indirect Optimal Control via Latent Geodesics (IG)*** *Comparison of performance with different settings of λ. The performance measures are summed over collocation points.*

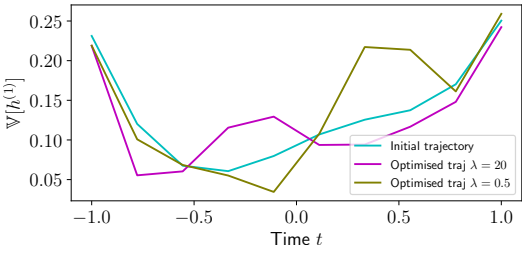
Trajectory	Mixing Probability $\sum_{i=1}^{I} \Pr(\alpha_i = 1 \mid \mathbf{x}_i)$	Epistemic Uncertainty $\sum_{i=1}^{I} \mathbb{V}[h_1(\mathbf{x}_i)]$
Initial	7.480	1.345
Optimised $\lambda = 20.0$	6.091	**1.274**
Optimised $\lambda = 0.5$	**8.118**	1.437

this trajectory favours remaining in the desired mode at the cost of entering the region of the gating network with high *epistemic uncertainty*. This is quantified in Table 5.1, which shows it accumulates more gating function variance than both the initial trajectory and the trajectory found with $\lambda = 20.0$.

In contrast, the trajectory found with $\lambda = 20.0$ initially remains in the desired mode but then enters the turbulent mode in favour of avoiding the area of high *epistemic uncertainty*. This is confirmed in Table 5.1 as the trajectory found with $\lambda = 20.0$ accumulates the least gating function variance over the trajectory. This is further visualised in Figures 5.3a and 5.3b. These results align with the intended behaviour of the user tunable λ parameter. However, in this experiment, increasing the relevance of the covariance term in the expected Riemannian metric has no benefit. This is because it pushes the trajectory into the turbulent dynamics mode.



(a) Desired mode's (α=1) mixing probability over the trajectories.



(b) Posterior variance associated with the desired mode's (α=1) gating function over the trajectories.

Figure 5.3: ***Indirect Optimal Control via Latent Geodesics (IG)*** *Comparision of the initial and optimised trajectories' performance – for two settings of λ – at a) staying in the desired mode and b) avoiding regions of the gating network with high epistemic uncertainty.*

These results indicate the nonlinear program in Equation (4.29), i.e. solving the geodesic ODE in Equation (4.27) via collocation, is capable of finding state trajectories from $\mathbf{x}_0$ to $\mathbf{x}_f$ that exhibit mode remaining behaviour. However, these experiments have not validated the method's ability to recover the controls from the state trajectory using Equation (4.33). This is because a full transition dynamics model has not been learned so the control trajectory cannot be recovered. In Section 5.2 the method is evaluated in a simulated environment where its ability to recover the controls is tested.

It is worth noting here that the mode remaining behaviour is sensitive to the tolerance on the collocation constraints. Setting the tolerance too small often resulted in the solver failing to converge, whilst setting the tolerance too large omitted any mode remaining behaviour.

5.2 Simulated Quadcopter Experiments

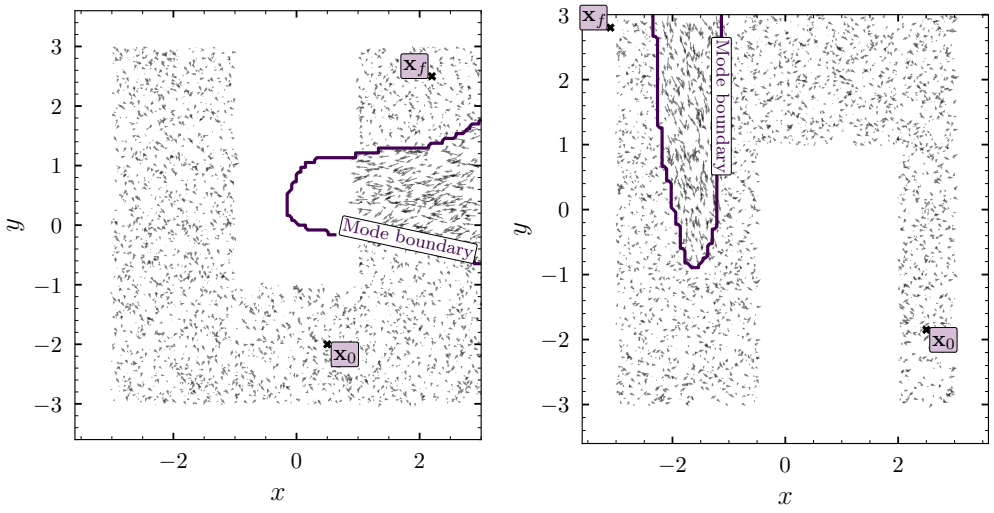
All of the control methods are now tested in two simulated environments so that they can be compared. This section first details the two simulation environments.

5.2.1 Simulator Setup

The simulated environments have two dynamics modes, $\alpha \in \{1, 2\}$, whose transition dynamics are given by a single integrator discretised using the forward Euler method (velocity times time). The modes are induced by different wind fields which are characterised by their drift $\boldsymbol{\omega}_k$ and process noise $\boldsymbol{\epsilon}_k$ terms. Each mode's dynamics are given by,

$$f_k(\mathbf{x}_t, \mathbf{u}_t; \Delta t = 0.25) = \mathbf{x}_t + \mathbf{u}_t \times \Delta t + \boldsymbol{\omega}_k + \boldsymbol{\epsilon}_k \tag{5.1}$$

$$\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_k}). \tag{5.2}$$



(a) ***Environment 1*** - $\boldsymbol{\omega}_1 = \{0.02, -0.4\}$, $\boldsymbol{\omega}_2 = \{-2.0, -0.4\}$, $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_1} = \text{diag}([0.0002, 0.0001])$ *and* $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_2} = \text{diag}([0.2, 0.05])$

(b) ***Environment 2*** - $\boldsymbol{\omega}_1 = \{0.4, 0.02\}$, $\boldsymbol{\omega}_2 = \{0.4, -2.0\}$, $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_1} = \text{diag}([0.0001, 0.0002])$ *and* $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_2} = \text{diag}([0.05, 0.2])$

Figure 5.4: ***Simulated Environments*** - *Visualisation of the simulated environments with their true mode boundaries indicated by the purple lines. The state transition data set sampled from each environment is visualised by the quiver plots. The initial* $\mathbf{x}_0$ *and target states* $\mathbf{x}_f$ *for the trajectory optimisation are overlayed.*

The state domain is constrained to $x, y \in [-3, 3]$ and min/max controls are implemented by constraining the control domain to $v_x, v_y \in [-5, 5]$.

Data set We sampled 4000 state transitions from each environment with $\Delta t = 0.25s$. We then removed a subset of the state transitions to induce a region of high *epistemic uncertainty* in the learned dynamics model. This enabled the methods' ability to avoid regions of high *epistemic uncertainty* to be tested. Figure 5.4 shows the two environments as well as the state transition data sets that were sampled from them. In Environment 2, the turbulent dynamics mode and the unobserved regions are in different locations to Environment 1. Further to this, the region with no observations does not overlap the mode boundary like in Environment 1.

5.2.2 Model Learning

Following the experiments in Section 5.1, the model from Chapter 3 was instantiated with $K = 2$ experts, one to represent the desired dynamics mode and one to

represent the turbulent dynamics mode. The experiments used the same GP priors and hyperparameters as the real-world experiments in Section 5.1. However, in contrast to the real-world experiments, the simulated experiments presented here learn the full transition dynamics model, i.e. they do not assume constant controls. Figures 5.5 and 5.6 show the gating network posteriors after training on the data sets from Environment 1 and Environment 2 respectively. In both environments, Expert 1 represents the turbulent dynamics mode and Expert 2 represents the operable, desired dynamics mode. This results from Expert 1 learning much higher drift and process noise than Expert 2.

5.2.3 Performance Indicators

Before evaluating the three trajectory optimisation algorithms in the simulated environments, let us restate the goals from Chapter 4:

Goal 1 Navigate to the target state $\mathbf{x}_f$,

Goal 2 Remain in the operable, desired dynamics mode k^*,

Goal 3 Avoid regions of the learned dynamics with high *epistemic uncertainty*,

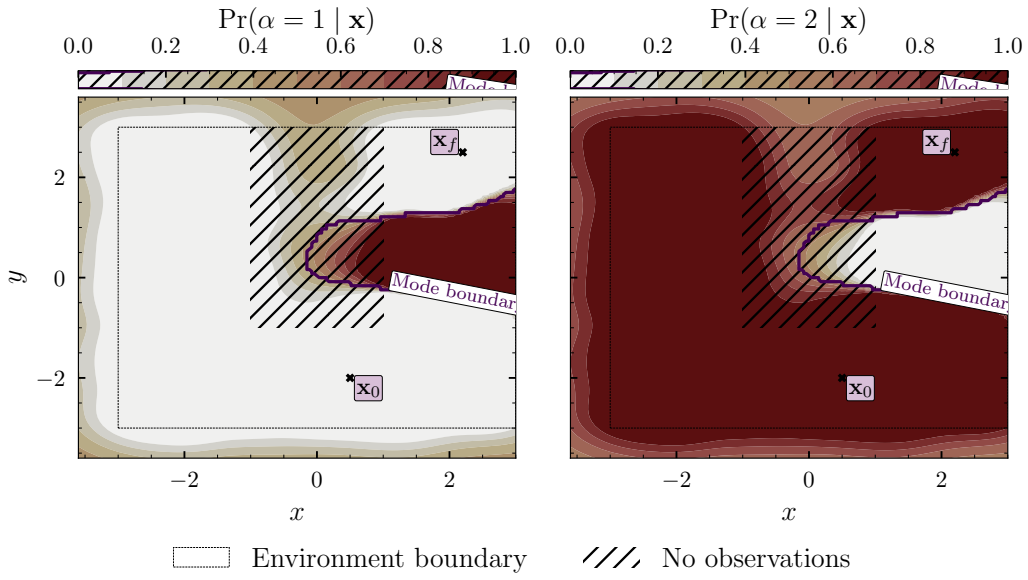
- **Goal 3.1** in the desired dynamics mode f_{k^*}, i.e. where the underlying dynamics are not known,
- **Goal 3.2** in the gating network α, i.e. where it is not known which mode governs the dynamics.

The performance of the trajectory optimisation algorithms are evaluated using four performance indicators:

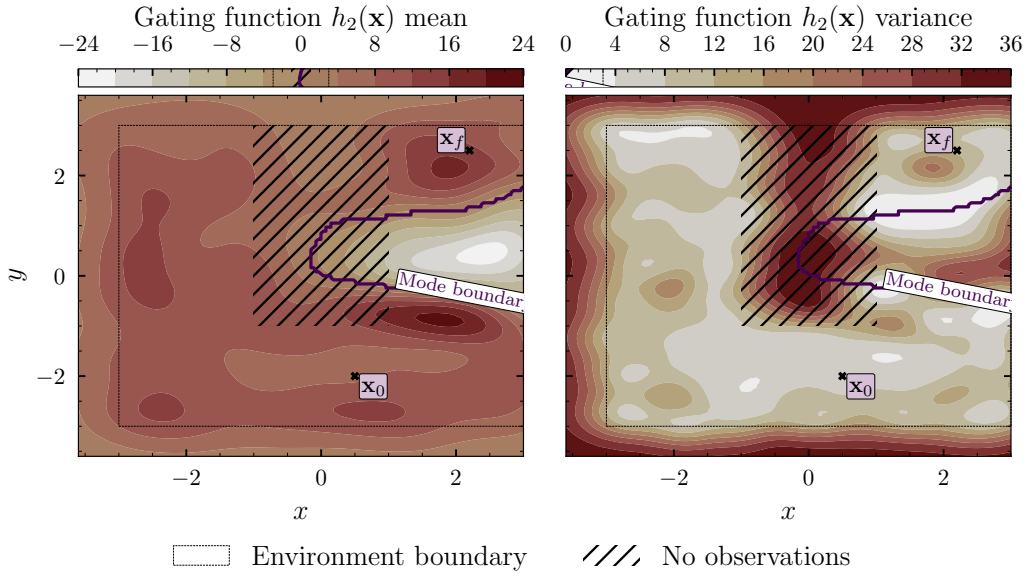
1. Goal 2 is measured using the probability of remaining in the desired mode without the model's uncertainty marginalised,

$$\sum_{t=0}^{T} \Pr(\alpha_t = k^* \mid h(\mathbf{x}_{0:t}), \mathbf{x}_{0:t}, \mathbf{u}_{0:t}, \alpha_{0:t-1} = k^*) \tag{5.3}$$

   This probability will only decrease when a trajectory leaves the desired mode.

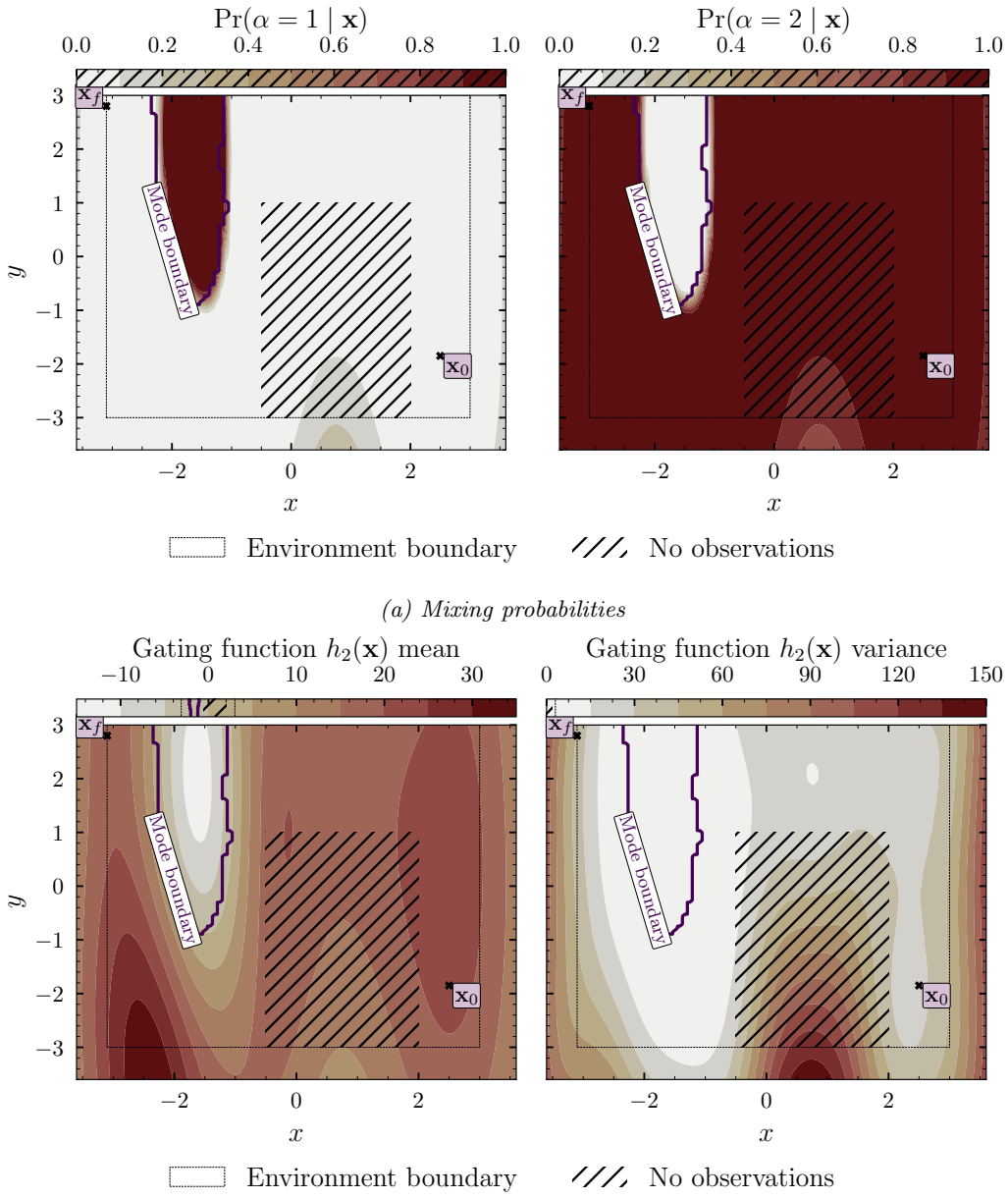


(a) Mixing probabilities



(b) Gating function's GP posterior

Figure 5.5: ***Environment 1*** *Visualisation of the gating network posterior after training MoSVGPE on the state transition data set from Environment 1. (a) shows the probability mass function over the expert indicator variable and (b) shows the gating function's GP posterior mean (left) and posterior variance (right). The start state* $\mathbf{x}_0$ *and target state* $\mathbf{x}_f$ *are overlayed along with the mode boundary (purple line) and the subset of the environment which has not been observed (hashed box).*



(a) Mixing probabilities

(b) Gating function's GP posterior

Figure 5.6: ***Environment 2*** *Visualisation of the gating network posterior after training MoSVGPE on the state transition data set from Environment 2. (a) shows the probability mass function over the expert indicator variable and (b) shows the gating function's GP posterior mean (left) and posterior variance (right). The start state $\mathbf{x}_0$ and target state $\mathbf{x}_f$ are overlayed along with the mode boundary (purple line) and the subset of the environment which has not been observed (hashed box).*

2. Goal 3.1 is measured using the state variance accumulated from cascading single-step predictions,

$$\mathbb{V}[\bar{\mathbf{x}}] = \sum_{t=1}^{T} \mathbb{V}[\mathbf{x}_t]. \tag{5.4}$$

   This will increase when a trajectory passes through regions of the desired dynamics mode with high *epistemic uncertainty*.

3. Goal 3.2 is measured using the gating function variance accumulated from cascading single-step predictions,

$$\mathbb{V}[h_{k^*}(\bar{\mathbf{x}})] = \sum_{t=1}^{T} \mathbb{V}[h_{k^*}(\mathbf{x}_t)]. \tag{5.5}$$

   This will increase when a trajectory passes through regions of the gating network with high *epistemic uncertainty*.

4. Probability of remaining in the desired mode with the model's uncertainty marginalised, calculated using Equation (4.42),

$$\sum_{t=1}^{T} \Pr(\alpha_t = k^* \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \alpha_{0:t-1} = k^*). \tag{5.6}$$

   This probability will decrease when a trajectory leaves the desired mode and when it passes through regions of the learned dynamics with high uncertainty.

The methods' ability to remain in the desired dynamics mode k^*, that is, to accomplish Goal 2, is measured using the probability of remaining in the desired dynamics mode Equation (5.3). The secondary goal of avoiding regions of the learned dynamics model with high *epistemic uncertainty* in the desired dynamics mode (Goal 3.1), is measured by summing the state variance over each trajectory Equation (5.4). Similarly, each method's ability to avoid regions of the gating network with high epistemic uncertainty in the gating network (Goal 3.2), is measured by summing the gating function variance over each trajectory Equation (5.5). Intuitively, the

Table 5.2: ***Results in simulated environments*** *Comparison of the Indirect Optimal Control via Latent Geodesics (IG) method from Section 4.2.2, the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3.2, with max entropy (MRCaI gauss) and without max entropy (MRCaI Dirac). All methods are evaluated in the two simulated environments. The performance measures are summed over the collocation points for the IG method and over each time step for the DRE method and the MRCaI method. The "Mode prob" column calculates the probability of being in the desired dynamics mode at each time step without marginalising the state and gating function uncertainty over the trajectory. In contrast the "Mode prob with unc." marginalises both the state and gating function uncertainty at each time step. The target state column indicates if the trajectory reached the target state in the environment (✓ both), only under the desired mode's dynamics GP (✓ dynamics) or not at all (×).*

Env	Method	λ	Mode prob Eq. 5.3	Mode prob with unc. Eq. 5.6	State unc. $\mathbb{V}[\bar{\mathbf{x}}]$	Gating unc. $\mathbb{V}[h_{k^*}(\bar{\mathbf{x}})]$	Target state $\mathbf{x}_T == \mathbf{x}_f$?
	IG	0.5	16.96	16.22	8.69	346.62	× dynamics
	IG	1.0	18.51	16.98	8.70	367.16	× dynamics
	IG	20.0	15.04	14.92	8.65	231.98	× dynamics
1	DRE	0.5	**20.98**	18.04	8.71	388.71	✓ both
	DRE	1.0	**20.98**	18.10	8.72	370.94	✓ both
	DRE	20.0	13.47	13.10	**8.64**	**164.88**	× dynamics
	MRCaI (gauss)	N/A	20.94	**19.99**	8.74	208.62	✓ both
	MRCaI (Dirac)	N/A	20.96	19.58	8.73	258.27	✓ both
	IG	0.5	18.98	19.22	1.53	495.11	×
	IG	1.0	18.98	19.16	1.39	522.60	×
	IG (mid point)	1.0	17.55	17.80	1.57	553.00	×
	IG	5.0	18.98	19.17	1.44	503.94	×
2	DRE	0.5	20.96	20.34	1.42	367.47	✓ both
	DRE	1.0	**20.98**	20.68	**1.28**	**334.47**	✓ both
	DRE	5.0	20.81	20.13	1.37	374.13	✓ both
	MRCaI (gauss)	N/A	**20.98**	**20.72**	1.75	596.67	✓ both
	MRCaI (Dirac)	N/A	**20.98**	20.61	1.56	644.35	✓ both

goal is to maximise the probability of being in the desired mode, whilst minimising the variance accumulated over a trajectory. This corresponds to maximising the probability of remaining in the desired dynamics mode with all of the model's *epistemic uncertainty* marginalised, i.e. maximising Equation (5.6).

5.2.4 Results

Three settings of the tunable λ parameter were tested for both of the geometry-based methods in each environment. The Mode Remaining Control as Inference (MRCaI) method from Section 4.3.2 was tested with Gaussian controls (gauss) and with deterministic (Dirac) controls. This enabled the maximum entropy control behaviour

(resulting from Gaussian controls) to be compared to a baseline without maximum entropy control. Table 5.2 summarises all of the results from both environments.

Initialisation All of the Direct Optimal Control via Riemannian Energy (DRE) and Mode Remaining Control as Inference (MRCaI) experiments used a horizon of $T = 20$ time steps and all of the Indirect Optimal Control via Latent Geodesics (IG) experiments were initialised with $I = 20$ collocation points. Further to this, all of the IG experiments were initialised with straight line trajectories between $\mathbf{x}_0$ and $\mathbf{x}_f$, except for one experiment in Environment 2, named IG λ=1.0 (mid point), which was initialised with two straight lines, between $\mathbf{x}_0$, $[-2.0, -2.0]$ and $\mathbf{x}_f$. This is because the IG experiments struggled to find solutions in Environment 2 – due to local optima – without adjusting the initial solution.

Visualisation Figures 5.8 to 5.10 show the trajectory optimisation results for both environments overlayed on their associated gating network posteriors. In each figure, the left-hand column shows results for Environment 1 whilst the right-hand column shows the results for Environment 2. The figures show the state trajectories obtained from cascading single-step predictions through the desired mode's GP dynamics using the controls found by the IG method (top row), the DRE method (middle row) and the MRCaI method (bottom row). Figure 5.8 shows the optimised trajectories overlayed on the desired mode's ($\alpha = 2$) mixing probability. This is useful for seeing how well trajectories remain in regions of the learned dynamics model with high probability of being in the desired mode. Figure 5.9 shows the trajectories overlayed on the gating function's posterior mean. As the geometry-based methods in this chapter are based on finding shortest trajectories on this manifold, the posterior mean plot is useful for observing contour following behaviour. As the methods should find trajectories that avoid regions of the gating network with high *epistemic uncertainty*, Figure 5.10 shows them overlayed on the gating function's posterior variance. Because the IG method is a two-stage process, which first finds a state trajectory using the collocation solver and then recovers the controls using the inference strategy in Equation (4.32), its collocation state trajectory is compared

to its dynamics trajectory in Figure 5.7. The three methods are now compared by analysing how well they achieve the three goals.

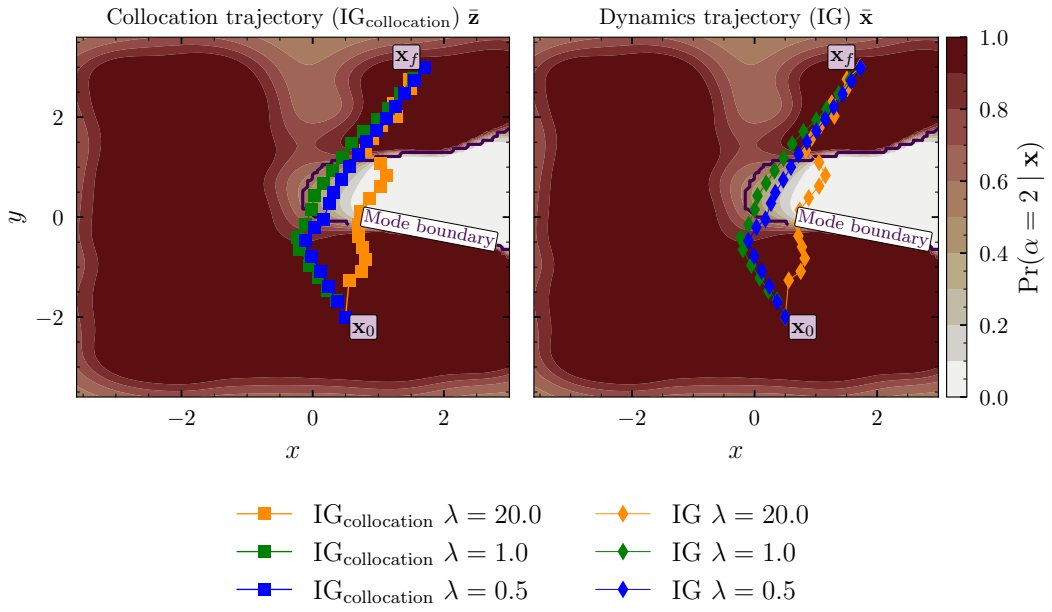
Goal 1 - Navigate to the Target State

The methods are first evaluated at their ability to navigate to the target state, that is, achieve Goal 1.

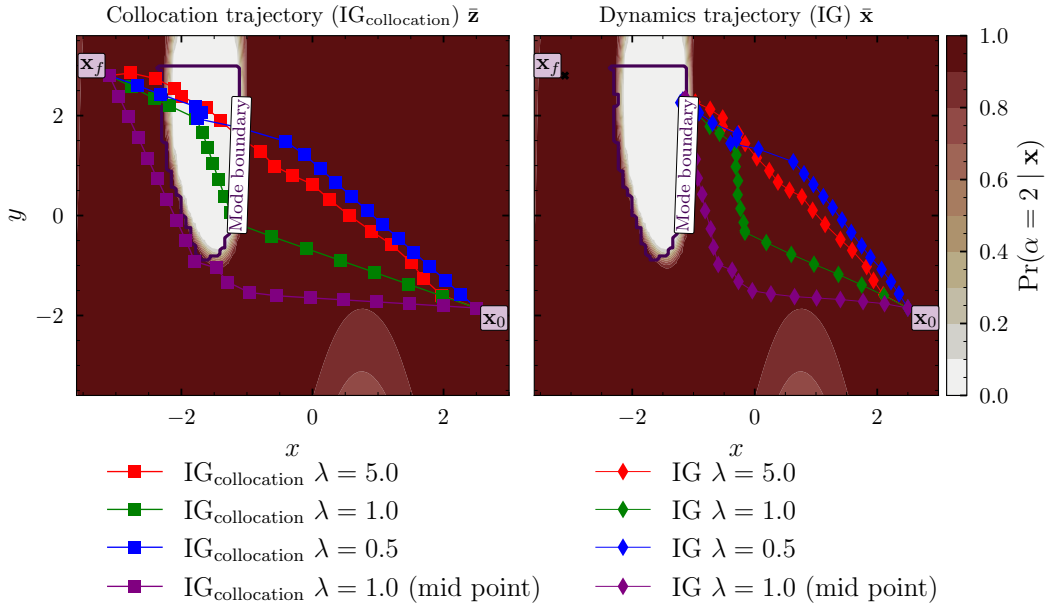
Indirect Optimal Control via Latent Geodesics (IG) The IG method's collocation solver was able to find state trajectories to the target state in all experiments. This is indicated by the collocation state trajectories $\bar{\mathbf{z}}$ (left-hand plots in Figure 5.7) reaching the target state $\mathbf{x}_f$ in both environments. This is because the collocation solver $\text{IG}_{\text{collocation}}$ is guaranteed to satisfy the boundary conditions. However, the inference strategy cannot always recover controls that drive the system along the collocation solver's state trajectory and thus to the target state $\mathbf{x}_f$.

In Environment 1, the inference strategy successfully recovered controls that drive the system along the collocation solver's state trajectory. This is indicated by the dynamics trajectory $\bar{\mathbf{x}}$ (right in Figure 5.7a), following the state trajectory $\bar{\mathbf{z}}$ found by the collocation solver $\text{IG}_{\text{collocation}}$ (left in Figure 5.7a). However, in Environment 2, the inference strategy could not recover the controls that could drive the system along the collocation solver's state trajectory. This can be seen by the dynamics trajectories (right in Figure 5.7b) differing from the collocation trajectories (left in Figure 5.7b.

One possible explanation is that the ELBO in Equation (4.32) could not recover the controls due to a local optimum. The collocation state trajectories (left) in the experiments with $\lambda = 0.5, 1.0, 5.0$, which were initialised with straight line trajectories, passed through the turbulent dynamics mode. As this region belongs to the turbulent dynamics mode, the desired mode's dynamics GP has high *epistemic uncertainty* in this region. This high uncertainty may have resulted in the ELBO being low when trajectories pass through this region and as a result, the gradient-based optimiser may have got stuck in the local optimum before this region. Further to this,



(a) Environment 1



(b) Environment 2

Figure 5.7: ***Indirect Optimal Control via Latent Geodesics (IG)*** *This figure shows how well the IG method's inference strategy in Equation (4.32) can recover the controls from the collocation solver's state trajectory $\bar{\mathbf{z}}$ in each environment. The left-hand plots show the state trajectories found by the collocation solver $\bar{\mathbf{z}}$. The right-hand plots show the trajectories $\bar{\mathbf{x}}$ obtained when rolling out the controls – recovered using the inference strategy in Equation (4.32) – in the desired mode's GP dynamics. They are overlayed on the desired mode's mixing probability for Environmnet 1 (top) and Environment 2 (bottom).*

the experiment that was initialised with a mid point at $[-2.0, -2.0]$ (purple) was not able to recover the correct controls, even though the collocation state trajectory (purple squares) does not pass through the turbulent dynamics mode. Monitoring the optimisation of the control trajectory showed that the control trajectory hit the local optima during optimisation. Therefore, it is possible that this experiment also got stuck in this local optimum. Although not tested, it may be possible to overcome this issue with random restarts, i.e. different initial control trajectories.

Direct Optimal Control via Riemannian Energy (DRE) All of the DRE experiments, (middle row of Figures 5.8 to 5.10) were able to find trajectories that navigate to the target state $\mathbf{x}_f$ under the desired mode's GP dynamics. It is worth noting that this method does not guarantee that trajectories will satisfy the boundary conditions. However, setting the terminal state cost matrix $\mathbf{H}$ to be very high, appears to work well.

Mode Remaining Control as Inference (MRCaI) All of the MRCaI experiments successfully navigated to the target state under the dynamics. This is indicated by the trajectories in the bottom row of Figures 5.8 to 5.10 successfully navigating to the target state $\mathbf{x}_f$. Similarly to the DRE method, this approach is not guaranteed to find trajectories that will end at the target state. However, setting a high terminal state cost matrix $\mathbf{H}$ worked well.

Goal 2 - Remain in the Desired Mode

The experiments are now evaluated at their ability to remain in the desired dynamics mode, that is, their ability to achieve Goal 2.

Indirect Optimal Control via Latent Geodesics (IG) None of the IG experiments were able to remain in the desired dynamics mode successfully. In all but one of the experiments, this was due to the collocation solver $\text{IG}_{\text{collocation}}$ finding trajectories that pass over the mode boundary into the turbulent dynamics mode. This is indicated by the collocation state trajectories in the left-hand plots of Fig-

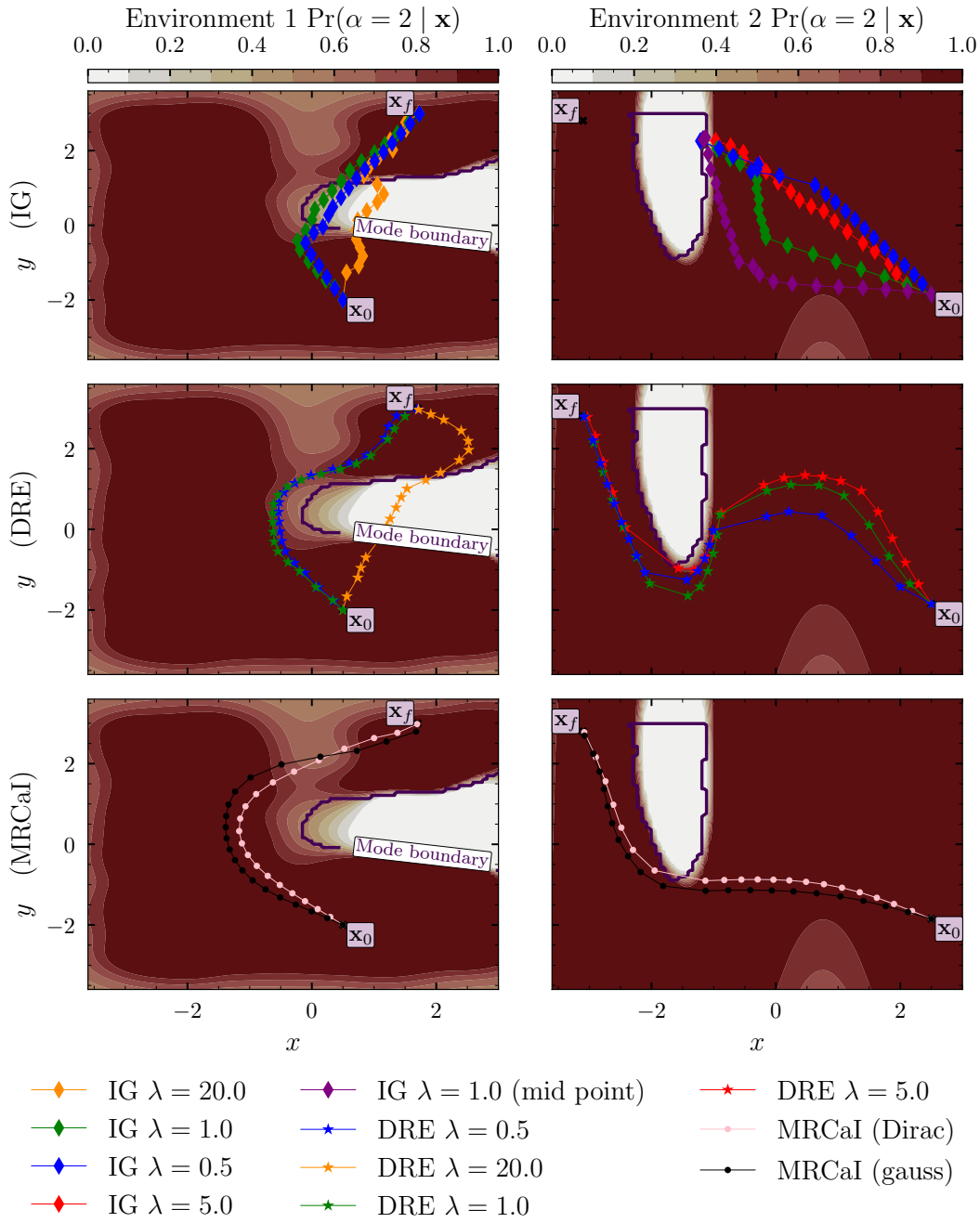


Figure 5.8: ***Trajectory optimisation results over the desired mode's mixing probability*** $\Pr(\alpha = k^* \mid \mathbf{x})$ *Trajectory optimisation results after finding trajectories from the start state* $\mathbf{x}_0$*, to the target state* $\mathbf{x}_f$*, using the Indirect Optimal Control via Latent Geodesics (IG) method from Section 4.2.2 (top row), the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 (middle row) and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3 (bottom row) in Environment 1 (left) and Environment 2 (right). The optimised controls are rolled out in the desired mode's GP dynamics and are overlayed on the desired mode's mixing probability.*

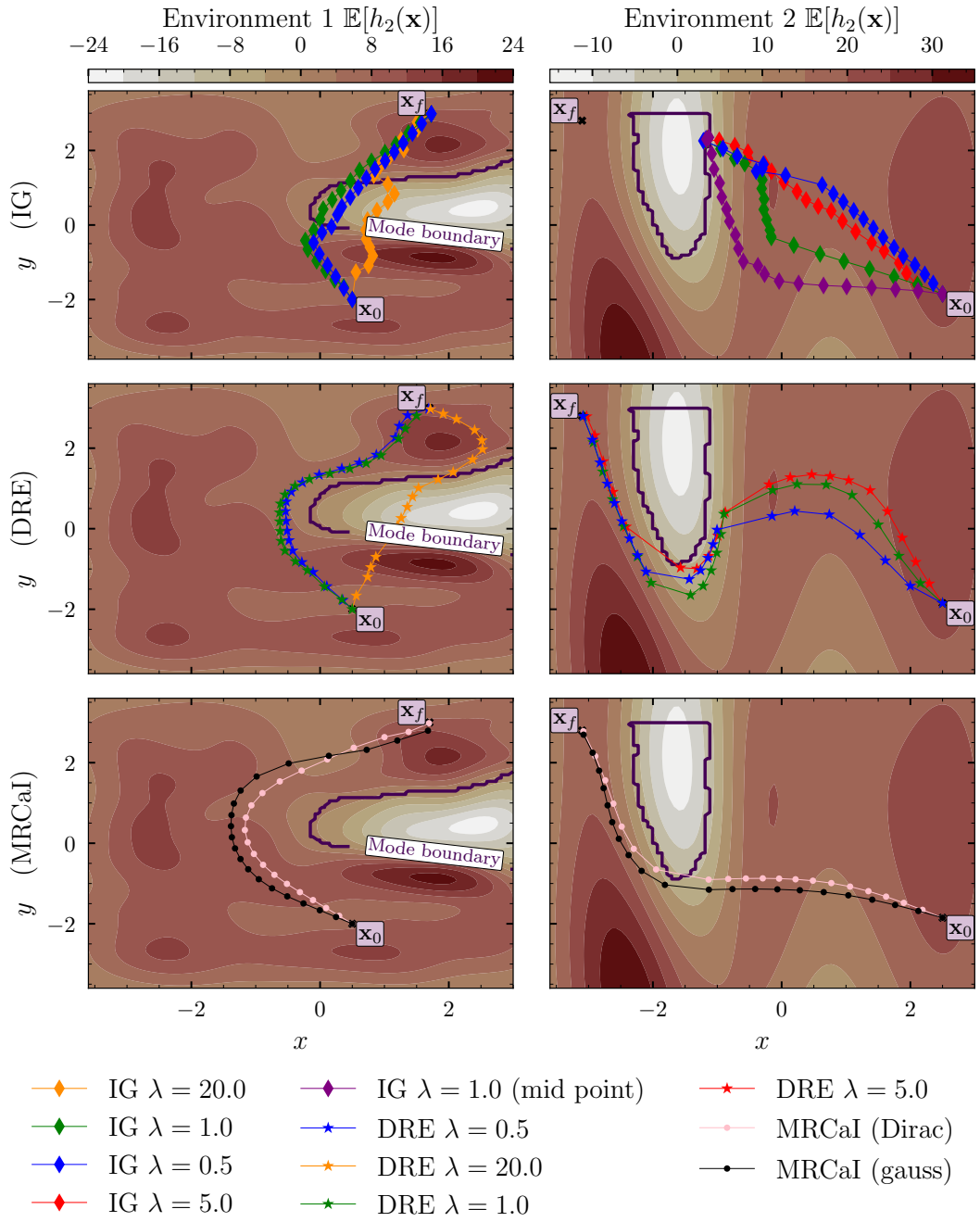


Figure 5.9: ***Trajectory optimisation results over the gating function's posterior mean*** $\mathbb{E}[h_{k^*}(\mathbf{x})]$ *Trajectory optimisation results after finding trajectories from the start state* $\mathbf{x}_0$*, to the target state* $\mathbf{x}_f$*, using the Indirect Optimal Control via Latent Geodesics (IG) method from Section 4.2.2 (top row), the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 (middle row) and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3 (bottom row) in Environment 1 (left) and Environment 2 (right). The optimised controls are rolled out in the desired mode's GP dynamics and are overlayed on the posterior mean associated with the desired mode's gating function.*

ure 5.7 passing directly through the turbulent dynamics mode. This motivated a further experiment, to see if the collocation solver was getting stuck in a local optimum induced by the initial trajectory. The $\text{IG}_{\text{collocation}}$ $\lambda = 1.0$ (mid point) experiment (purple squares in Figure 5.7b) was initialised as two straight lines with a mid point at $[-2.0, -2.0]$ to see if setting the initial trajectory not to pass over the mode boundary could enable it to find a mode remaining trajectory. Figure 5.7b indicates that with this initialisation, the collocation solver was able to find a mode remaining trajectory. This result indicates that the collocation solver is susceptible to local optima induced by the initial trajectory. Note that the need to initialise the collocation solver in this way limits the method's applicability. For example, how should the initial solution be set in environments where the state-space cannot be visualised as easily?

In Environment 1, although none of the IG experiments successfully remained in the desired dynamics mode, they did exhibit some mode remaining behaviour. That is, the trajectories with $\lambda = 0.5$ (blue) and with $\lambda = 1.0$ (green) in Figure 5.7a navigate to the left and almost reach the mode boundary. However, they could not find solutions that remained in the desired mode for the entire trajectory.

Direct Optimal Control via Riemannian Energy (DRE) All but one of the DRE experiments were able to remain in the desired dynamics mode successfully. This is indicated by the dynamics trajectories in the middle row of Figures 5.8 to 5.10 not passing over the mode boundary into the turbulent dynamics mode. From visual inspection of the middle rows in Figures 5.8 to 5.10 the trajectories found with DRE $\lambda = 1.0$ (green stars) have the most clearance from the turbulent dynamics in both environments, compared to other DRE λ experiments. The trajectories in the middle row of Figure 5.9 show the contour following that achieves the mode remaining behaviour. In particular, the second half of the Environment 2 trajectories in the right-hand plot. Table 5.2 confirms that in both environments, the trajectory found with DRE $\lambda = 1.0$ obtained the highest probability of remaining in the desired dynamics mode when not considering the model's epistemic uncertainty.

Although the trajectories found with DRE $\lambda = 0.5, 1.0$ in Environment 1 obtained the highest probability when not considering the *epistemic uncertainty*, they did not obtain the highest probabilities when considering it. This is because they both accumulate high gating function variance when they pass over the region of high *epistemic uncertainty* in the gating network. This indicates that it is important to consider the model's *epistemic uncertainty* when finding mode remaining trajectories.

The trajectory found with DRE $\lambda = 20.0$ in Environment 1 obtained the lowest probability of remaining in the desired dynamics mode. Note that increasing the relevance of the covariance term (increasing λ) is equivalent to decreasing the relevance of the mode remaining term. As a result, the optimisation landscape has favoured trajectories that avoid the region of high posterior variance in the gating network more than following a true geodesic path on the mean of the gating function. This can be seen in Figure 5.10 where the trajectory found with DRE $\lambda = 20.0$ (orange stars) did not successfully remain in the desired dynamics mode. This indicates that setting the relevance of the covariance term too high can have a negative impact on performance. These results suggest that care should be taken when adjusting the value of λ.

It is worth noting that in practice the mode chance constraints would inform us that the trajectory is not $\delta -$ mode remaining, so the trajectory would not be executed in the environment.

Mode Remaining Control as Inference (MRCaI) It is clear from the bottom rows of Figures 5.8 to 5.10 that all MRCaI experiments found trajectories that remained in the desired dynamics mode. Further to this, the maximum entropy control term resulted in more clearance from the mode boundary. This is shown by the experiments with Gaussian controls MRCaI (gauss), indicated by black circles, having more clearance than the experiments with deterministic controls MRCaI (Dirac), indicated by the pink circles. This is a desirable behaviour that is expected from the maximum entropy control term. This observation is confirmed in Table 5.2, where

in both environments the experiments with maximum entropy control obtained the highest probabilities of remaining in the desired dynamics mode when considering the model's *epistemic uncertainty*.

In Environment 2, both of the MRCaI experiments found discrete-time trajectories that remained in the desired dynamics mode. This is indicated by none of the trajectories' time steps being in the turbulent dynamics mode. However, when interpolating the discrete-time trajectory for the Environment 2 experiment without maximum entropy control (pink circles), shown in the bottom right plots in Figures 5.8 to 5.10, the trajectory crosses the mode boundary. This is an undesirable behaviour because in non-simulated environments this would correspond to the trajectory entering the turbulent dynamics mode. In contrast, the clearance resulting from the maximum entropy control term alleviates this issue.

Goal 3 - Avoid Regions of High Epistemic Uncertainty

Finally, the methods are evaluated at their ability to avoid regions of the dynamics with high *epistemic uncertainty*. Table 5.2 shows the state variance and the gating function variance accumulated over each trajectory. In all Environment 1 experiments, the state variance accumulated over the trajectory (from cascading single-step predictions via moment matching) were fairly similar. This is due to the dynamics of the simulator being simple enough that the desired mode's GP can confidently interpolate into both the turbulent dynamics mode and the region which has not been observed. In the latter case, it is up to the gating network to model the *epistemic uncertainty* arising from limited training data.

In contrast to the experiments in Environment 1, the state variance accumulated over each trajectory does vary between experiments in Environment 2. However, the results in Table 5.2 suggest that the state variance does not significantly impact the mode remaining behaviour. This is indicated by no correlation between the mode probability and the state variance, which is likely due to extremely low state variance.

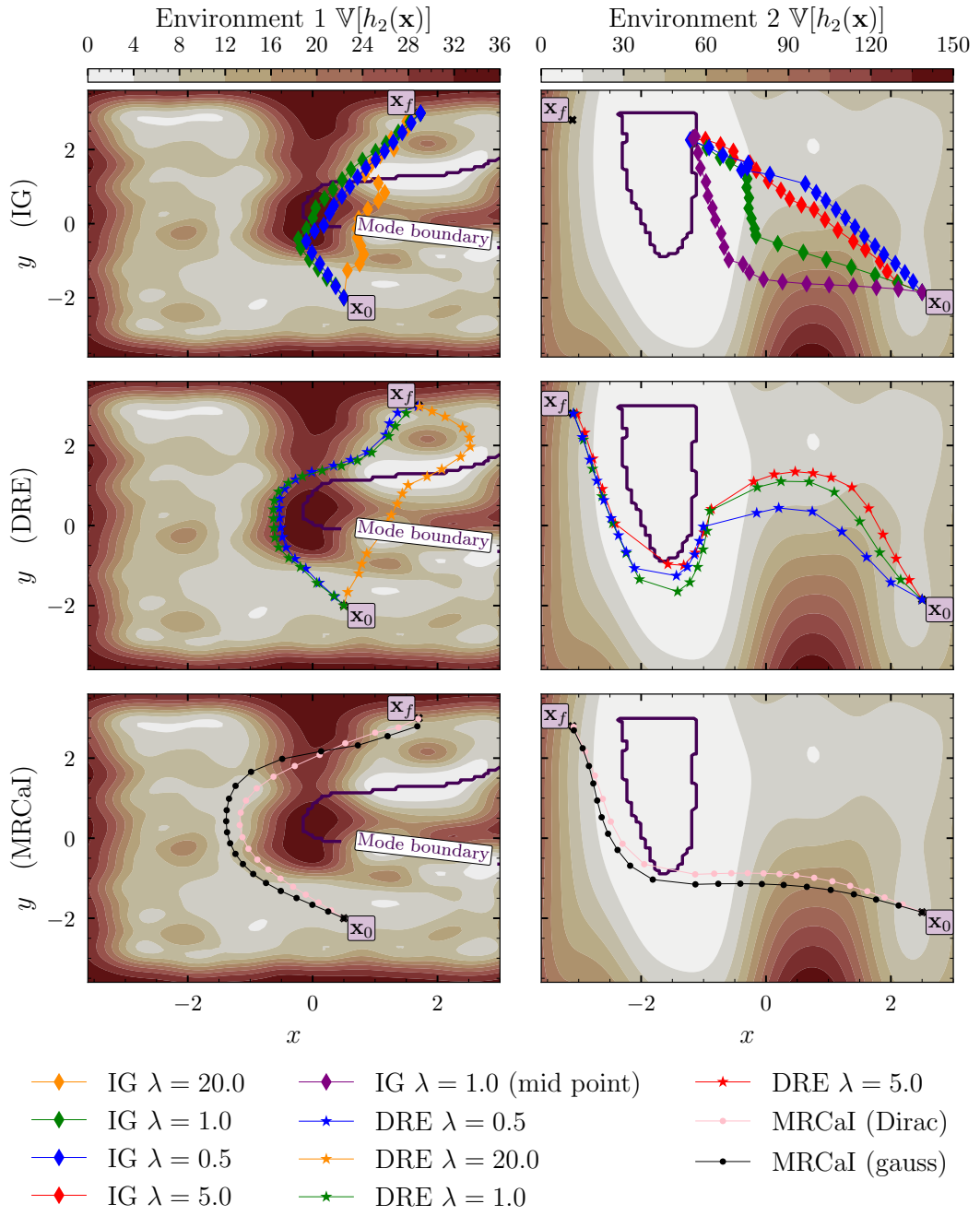


Figure 5.10: ***Trajectory optimisation results over the gating function's posterior variance*** $\mathbb{V}[h_{k^*}(\mathbf{x})]$ *Trajectory optimisation results after finding trajectories from the start state* $\mathbf{x}_0$*, to the target state* $\mathbf{x}_f$*, using the Indirect Optimal Control via Latent Geodesics (IG) method from Section 4.2.2 (top row), the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 (middle row) and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3 (bottom row) in Environment 1 (left) and Environment 2 (right). The optimised controls are rolled out in the desired mode's GP dynamics and are overlayed on the posterior variance associated with the desired mode's gating function.*

Indirect Optimal Control via Latent Geodesics (IG) In the IG experiments in Environment 1 with $\lambda = 0.5, 1.0$, the trajectories navigate directly over the region of high posterior variance associated with the gating function. This can be seen in the top left plot in Figure 5.10, by the blue and green diamonds trajectories. In contrast, the IG experiment with $\lambda = 20.0$ (orange diamonds) avoided this region. However, this uncertainty avoiding behaviour came at the cost of passing straight through the turbulent dynamics mode. In Environment 2 the IG experiments did not remain in the desired dynamics mode, nor did they exhibit any uncertainty avoiding behaviour. As mentioned previously, this was likely due to the collocation solver getting stuck in local optima as well as the inference strategy struggling to recover the correct controls.

Direct Optimal Control via Riemannian Energy (DRE) The DRE experiments exhibited the most uncertainty avoiding behaviour. This is indicated in Table 5.2 by the Environment 1 experiment with DRE $\lambda = 20.0$ obtaining the lowest accumulation of gating function variance out of all the experiments. The middle left plot in Figure 5.10 shows that the trajectory found with DRE $\lambda = 20.0$ (orange stars) avoids the region of high gating function variance. However, similarly to the IG experiments, this came at the cost of leaving the desired dynamics mode. These results confirm that the covariance term in the expected Riemannian metric can encode the notion of avoiding regions of high *epistemic uncertainty* in the gating network. However, it also suggests that adjusting λ to avoid regions of high *epistemic uncertainty* in the gating network is not always favourable and may result in the optimisation landscape favouring trajectories that leave the desired dynamics mode; essentially defeating the goal of introducing the λ parameter in the first place.

In Environment 2 the trajectory found with DRE $\lambda = 1.0$ accumulated the least state and gating function variance. Interestingly, Table 5.2 indicates that the experiment

with DRE $\lambda = 5.0$ resulted in less uncertainty avoiding behaviour. Further to this, the DRE $\lambda = 1.0$ experiment, which accumulated the least amount of state and gating function variance obtained the second highest probability of remaining in the desired dynamics mode when considering the model's *epistemic uncertainty*. This result suggests that avoiding regions of high *epistemic uncertainty* is favourable for remaining in the desired dynamics mode. However, when combined with the result in Environment 1 it is not possible to draw this conclusion. Not only did increasing the value of λ not always lead to an increase in the uncertainty avoiding behaviour, but increasing the uncertainty avoiding behaviour was also not always beneficial, as indicated by the Environment 1 results with DRE $\lambda = 20.0$.

These results indicate that the covariance term – in the expected Riemannian metric – plays an important role at keeping trajectories in the desired dynamics mode. However, they also indicate that λ alters both the mode remaining term and the covariance term, in a complex manner. As such, it is not always clear how the value of λ should be set. This is especially the case in Environment 1, where the inoperable dynamics mode intersects the region of high *epistemic uncertainty*. In this case, the effect of adjusting the tunable λ parameter is complex enough that it may be best to just leave it at $\lambda = 1.0$.

Mode Remaining Control as Inference (MRCaI) The MRCaI experiments in Environment 1 avoided the region of high *epistemic uncertainty* in the gating network. This is shown by the MRCaI (gauss) (black circles) and the MRCaI (Dirac) (pink circles) in the bottom left plot of Figure 5.10, navigating far to the left around the region of high gating function variance. This uncertainty avoiding behaviour is confirmed in Table 5.2, where the MRCaI experiments accumulated the least amount of gating function variance out of all the trajectories which remained in the desired dynamics mode. However, in Environment 2, the MRCaI experiments did not exhibit as much uncertainty avoiding behaviour in the gating network. In fact, they obtained higher accumulations of gating function variance by quite a margin. As mentioned earlier, this result suggests that avoiding regions of high *epistemic*

uncertainty is not the most important factor for obtaining the highest probability of remaining in the desired dynamics mode. This change in behaviour is due to the relevance of the uncertainty avoiding behaviour being automatically handled by the marginalisation of the gating function in Equation (4.63).

Although marginalisation is a principled way of handling uncertainty, in this scenario, it can be conjectured otherwise. First, note that the quantitative performance measures do not tell the full story. This is because in the region with no observations, it is perfectly plausible that there is another region belonging to the turbulent dynamics mode, or a different dynamics mode altogether. In this case, the trajectory found with DRE $\lambda = 5.0$ (red stars) is most likely to avoid entering the turbulent dynamics mode, because it avoids the region of high *epistemic uncertainty* associated with no observations. The probability of remaining in the desired dynamics mode does not capture this notion. There are two reasons for this. Firstly, the gating network is overconfident in this region due to learning a long lengthscale. This could be overcome by fixing the lengthscale a priori. However, there is a second issue due to the fundamental modelling approximation made by GPs. The GP-based gating network is not aware that the region with no observations is so large that it could fit another turbulent dynamics mode inside it. This is due to the Squared Exponential kernel function relying on point-wise calculations to build the covariance matrix. Future work could explore methods that capture higher-order interactions, for example, the size and shape of modes.

In both environments, the MRCaI (guass) experiments with maximum entropy control obtained lower accumulations of gating function variance than the MRCaI (Dirac) experiments without maximum entropy control. This further confirms that the maximum entropy control behaviour improves the performance of the MRCaI method.

5.3 Conclusion

This section details the main findings from the experiments, compares the methods and discusses directions for future work.

Geometry-based methods The IG method's collocation solver is susceptible to local optima around its initial state trajectory. Although this can be overcome by engineering the initial solution, it makes it difficult to get the method working in practice. On top of this, the variational inference strategy used to recover the controls does not always recover the correct controls. This may be due to the collocation solver's state trajectory not satisfying the dynamics constraints or the variational inference getting stuck in local optima. Regardless, this is a big limitation that made the IG method fail in some environments.

Overall, the DRE method performed significantly better than the IG method in the experiments. Firstly, two of the DRE experiments in Environment 1 remained in the desired dynamics mode, whilst the IG experiments could not. Further to this, the DRE approach worked in Environment 2, where it was not possible to get the IG method working, without engineering the initial solution. Not only did the DRE approach perform better in the experiments, but it is also significantly easier to configure. That is, setting the optimisation parameters is straightforward. In contrast, setting the parameters for the IG method is not. In particular, setting the upper and lower bounds for the collocation constraints is pernickety. In conclusion, the DRE method is the preferred geometry-based method for finding mode remaining trajectories.

How to set λ? Although increasing λ generally leads to more uncertainty avoiding behaviour in the gating network, this is not necessarily the case. Further to this, avoiding regions of high *epistemic uncertainty* in the gating network does not always lead to higher probabilities of remaining in the desired dynamics mode under Equation (5.6). As a result, care should be taken if/when adjusting λ.

The quantitative results indicate that good performance is generally achieved with $\lambda = 1.0$. That is, not modifying the expected Riemannian metric tensor. Although some benefits can be obtained via setting λ, for example, encoding a notion of risk-sensitivity for avoiding the region with no observations, it is not always clear how it should be set. Further to this, in many realistic scenarios, the state-space will not be easy to visualise like in the 2D quadcopter experiments. For these reasons, I conclude that it is best to air on the side of caution when setting λ. That is, λ should remain at 1.0 unless it is immediately clear how it should be set.

Mode Remaining Control as Inference (MRCaI) The MRCaI method performed well in all experiments. The experiments showed that the maximum entropy control behaviour obtained from using normally distributed controls improves the mode remaining behaviour and works well in practice.

Overall The results in Table 5.2 indicate that the MRCaI (gauss) experiments were the highest performing in both environments. They obtained the highest probability of remaining in the desired dynamics mode when considering the model's epistemic uncertainty. Visual inspection combined with the results in Table 5.2, further indicate that the DRE $\lambda = 1.0$ experiments performed well in both environments. Not only did the DRE experiments avoid leaving the desired dynamics mode but they also avoided the region of the gating network with high *epistemic uncertainty* more so than the MRCaI experiments. Both the Direct Optimal Control via Riemannian Energy (DRE) and the Mode Remaining Control as Inference (MRCaI) with maximum entropy control MRCaI (gauss) methods are competitive approaches for finding mode remaining trajectories.

5.3.1 Discussion & Future Work

This section compares the three control methods presented in Sections 4.2.2, 4.2.3 and 4.3 and suggests future work to address their limitations. Table 5.3 offers a succinct comparison of the three approaches.

Table 5.3: *Comparison of the Indirect Optimal Control via Latent Geodesics (IG) method from Section 4.2.2, the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3.*

	MRCaI	IG	DRE
Dynamics constraints guaranteed?	✓	×	✓
Considers *epistemic uncertainty* in dynamics?	✓	×	✓
Considers *epistemic uncertainty* in gating network?	✓	✓	✓
Can remain in **multiple** modes?	✓	×	×
Boundary conditions guaranteed?	×	✓	×
$\delta -$ mode remaining?	✓	×	✓
Continuous-time trajectory?	×	✓	×

Dynamics constraints Both the MRCaI method presented in Section 4.3 and the DRE approach presented in Section 4.2.3 find trajectories that satisfy the dynamics constraints, i.e. the learned dynamics. They achieve this by enforcing the distribution over state trajectories to match the distribution from cascading single-step predictions through the desired mode's dynamics GP. This can be seen as a method for approximating the integration of the controlled dynamics with respect to time, whilst considering the *epistemic uncertainty* associated with learning from observations. The chance constraints ensure the controlled system is $\delta -$ mode remaining, which makes the approximate dynamics integration valid. In contrast, the IG method from Section 4.2.2 does not find trajectories that are guaranteed to satisfy the dynamics constraints. This is a limiting factor that makes the IG method less appealing. Therefore, methods for incorporating the dynamics constraints into IG are an interesting direction for future work.

Decision-making under uncertainty The combination of the approximate dynamics integration and the chance constraints in the DRE method, leads to a closed-form expression for the expected cost. This expression considers the *epistemic uncertainty* associated with the learned dynamics model, both in the desired dynamics mode and in the gating network. Similarly, for the MRCaI method, the ELBO principally handles the uncertainty in both the desired dynamics mode and gat-

ing network. In contrast, the IG method ignores the uncertainty accumulated by cascading single-step predictions through the learned dynamics model. That is, it considers the uncertainty associated with the gating network and ignores any uncertainty in the desired dynamics mode. Although this did not have a massive impact in the simulated quadcopter experiments, it may have a larger impact in real-world systems with more complex dynamics modes.

Uncertainty propagation This work only considered the moment matching approximation for propagating uncertainty through the probabilistic dynamics model. Chua et al., 2018 test different uncertainty propagation schemes in Bayesian neural networks. It would be interesting to test sampling-based approaches for the DRE approach. For example, to see the impact of calculating the expected Riemannian energy over a trajectory without approximations.

Decouple goals Without decoupling the uncertainty avoiding behaviour from the mode remaining behaviour, it is not possible to find trajectories which avoid entering the region of high *epistemic uncertainty* in the gating network. The geometry-based approaches provide a mechanism to set the relevance of the covariance term, i.e. decouple the goals. This was achieved by augmenting the expected Riemannian metric tensor with the user-tunable weight parameter λ, which can be adjusted to determine the relevance of the covariance term. The experiments show that although adjusting λ can be beneficial in some scenarios, it is not necessarily straightforward to set. In particular, when regions of high *epistemic uncertainty* intersect with mode boundaries. Therefore, care should be taken when setting λ. The MRCaI method does not provide a mechanism for adjusting the relevance of each behaviour. Instead, it relies on the marginalisation of the latent variables in the ELBO (Equation (4.64)) to automatically handle it, without requiring human input. This works well in practice but may be a limitation if it is desirable to decouple the goals and balance the mode remaining and uncertainty avoiding behaviour.

Multiple desired modes Although not tested, the approaches are theoretically sound and should be applicable in systems with more than two dynamics modes.

However, it is interesting to consider if this is even necessary given the goals. For example, in the quadcopter experiment, the dynamics model was intentionally instantiated with two dynamics modes, even though there could be more in reality. The desired dynamics mode was engineered to have a noise variance that was deemed operable. The other dynamics mode was then used to explain away all of the inoperable modes. In most scenarios, a similar approach could be followed. Nevertheless, it is interesting to consider systems with more than two modes. This is because the main goal is to avoid entering the inoperable dynamics mode, not just remain in the desired dynamics mode. Although not tested here, the MRCaI method should be able to remain in more than one desired dynamics mode. This can be achieved by conditioning on a set of modes. In contrast, the geometry-based methods in Section 4.2 are only capable of remaining in a single dynamics mode.

Continuous-time trajectories The DRE and MRCaI methods required control regularisation to prevent state transitions that "jump" over the undesired mode. Although the discrete-time steps of the trajectory appear to satisfy the constraints and minimise the cost, in reality, the continuous-time trajectory may pass through the undesired mode. This is a general problem that arises when solving continuous-time problems in discrete time. In contrast, the IG method uses a collocation solver where the cost can be evaluated at arbitrary points along the continuous-time trajectory. An interesting direction for future work is to extend the DRE and MRCaI methods to find trajectories in continuous time. For example, Dong et al., 2016; Mukadam et al., 2018 use GPs for continuous-time trajectories when solving motion planning via probabilistic inference.

State-control dependent modes This chapter assumed that the dynamics modes were separated according to their disjoint state domains, i.e. $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for distinct $i, j \in \{1, \ldots, K\}$. It would be interesting to extend this work to systems where the modes are governed by both state and control. For example, flying at high speed through a wind field may be deemed an operable dynamics mode, whilst flying at low speed may not.

Dynamics models The MRCaI method can be deployed in a wider range of dynamics models than the geometry-based methods. First of all, it is not limited to differentiable mean and covariance functions for the gating function GPs. Secondly, it can be deployed in dynamics models learned with any MoGPE method. This is because all MoGPE methods consist of a probability mass function over the expert indicator variable. In contrast, the geometry-based methods are limited to the MoSVGPE method because they depend on it's GP-based gating network. Further to this, the MRCaI method is applicable in systems where the dynamics modes are not necessarily separated according to their state domains $\mathcal{X}_k$. This is because it does not rely on the state-dependent gating functions.

Real-time control Whilst real-time control requires efficient algorithms, "offline" trajectory optimisation can trade in computational cost for greater accuracy. This work is primarily interested in finding trajectories that attempt to remain in the desired dynamics mode. For simplicity, it has considered the "offline" trajectory optimisation setting. The increased computational time may hinder its suitability to obtain a closed-loop controller via MPC (Eduardo F. and Carlos, 2007). However, it can be used "offline" to generate reference trajectories for a tracking controller or for guided policy search in a MBRL setting (Levine and Koltun, 2013). Alternatively, future work could investigate approximate inference methods for efficient state estimation to aid with real-time control, e.g. iLQG/GP.

Infeasible trajectories It is worth noting that it might not be possible to find a trajectory between a given start state $\mathbf{x}_0$ and a target state $\mathbf{x}_f$, that satisfies the chance constraints. This may be due to either the desired dynamics mode being uncertain, or the gating network being uncertain. In this scenario, it is desirable to explore the environment and update the learned dynamics model with this new experience. This will reduce the *epistemic uncertainty* in the model, increasing the likelihood of being able to find trajectories that satisfy the chance constraints. This motivates the work in Chapter 6, which addresses exploration in multimodal dynamical systems, whilst attempting to remain in the desired dynamics mode.

5.3.2 Summary

This chapter has evaluated and compared the Indirect Optimal Control via Latent Geodesics (IG) method from Section 4.2.2, the Direct Optimal Control via Riemannian Energy (DRE) method from Section 4.2.3 and the Mode Remaining Control as Inference (MRCaI) method from Section 4.3. The methods' abilities to navigate to a target state, whilst remaining in the desired dynamics mode, were evaluated on two velocity controlled quadcopter navigation problems. The results in this chapter have verified that the latent geometry of the MoSVGPE gating network can be used to encode mode remaining behaviour into control strategies. However, the results indicate that the IG method is not only the lowest performing method but is also the hardest method to configure. In contrast, the DRE and MRCaI methods work well in practice, whilst being much easier to configure.

Although the geometry-based methods have a tunable parameter λ for balancing the mode remaining and the uncertainty avoiding behaviour, it does not work well in practice. In contrast, the MRCaI method automatically balances the behaviours by marginalising the uncertainty in it's ELBO.

From the experiments in this chapter, it can be concluded that both the DRE and MRCaI methods are competitive approaches for finding mode remaining trajectories. They find trajectories that navigate to the target state, satisfy the dynamics constraints and attempt to remain in the desired dynamics mode. Further to this, it is easy to verify that they are $\delta-$mode remaining by evaluating the mode chance constraints.

6 Mode Remaining Exploration for Model-Based Reinforcement Learning

> Real knowledge is to know the extent of one's ignorance."
>
> *Confucius (Philosopher, 551–479BC).*

Similarly to Chapter 4, this chapter is concerned with controlling *unknown* or *partially unknown*, multimodal dynamical systems, from an initial state $\mathbf{x}_0$ – in the desired dynamics mode k^* – to a target state $\mathbf{x}_f$, whilst guaranteeing that the controlled system remains in the desired dynamics mode. However, in contrast to previous chapters, it does not assume prior access to the environment. That is, it considers the more realistic scenario, where the agent must iteratively explore its environment, collect data and update it's dynamics model – whilst remaining in the desired dynamics mode – until it can confidently navigate to the target state $\mathbf{x}_f$. Following previous chapters, this chapter also considers model-based approaches where the dynamics model is learned from observations.

The main contribution of this chapter is an explorative trajectory optimisation algorithm that can explore multimodal environments with a high probability of remaining in the desired dynamics mode. This chapter further proposes an approach

to solve the mode remaining navigation problem from Section 2.1 by consolidating all of the work in this thesis.

This chapter is organised as follows. Section 6.1 formally states the problem and the assumptions that are made in this chapter. Section 6.2.1 then details the exploration strategy proposed in this work and Section 6.2.2 presents Mode Optimisation (ModeOpt), a MBRL algorithm for solving the mode remaining navigation problem in Section 2.1. Finally, Section 6.3 presents preliminary results in a simulated quadcopter environment and Section 6.4 discusses ModeOpt and details directions for future work.

6.1 Problem Statement

Similarly to Chapter 4, the goal of this chapter is to solve the mode remaining navigation problem in Section 2.1. That is, to navigate from an initial state $\mathbf{x}_0$ – in the desired dynamics mode k^* – to the target state $\mathbf{x}_f$, whilst guaranteeing that the controlled system remains in the desired dynamics mode. However, this chapter does not assume *a priori* access to the environment, such that a data set of state transitions can be collected and used to learn a dynamics model. Instead, the agent must incrementally explore the environment without violating the mode remaining constraints in Definition 2.1.1.

Given that this work leverages a learned dynamics model, it is not possible to find trajectories that are mode remaining according to Definition 2.1.1. Following Chap-

ter 4, this work relaxes the mode remaining constraint to be $\delta-$ mode remaining, i.e. mode remaining with high probability. Formally this problem is given by,

$$\min_{\pi \in \Pi} \quad J_\pi(\mathbf{x}_0) \tag{6.1a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f_k(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) + \epsilon_k, \quad \alpha(\mathbf{x}_t) = k \qquad \forall t \in \{0, \dots, T-1\} \tag{6.1b}$$

$$\Pr(f(\mathbf{x}_t, \pi(\mathbf{x}_t, t)) \in \mathcal{X}_{k^*}) \geq 1 - \delta \qquad \forall t \in \{0, \dots, T-1\} \tag{6.1c}$$

$$\pi(\mathbf{x}_t, t) \in \mathcal{U} \qquad \forall t \in \{0, \dots, T-1\} \tag{6.1d}$$

$$\mathbf{x}_0 = \mathbf{x}_0 \tag{6.1e}$$

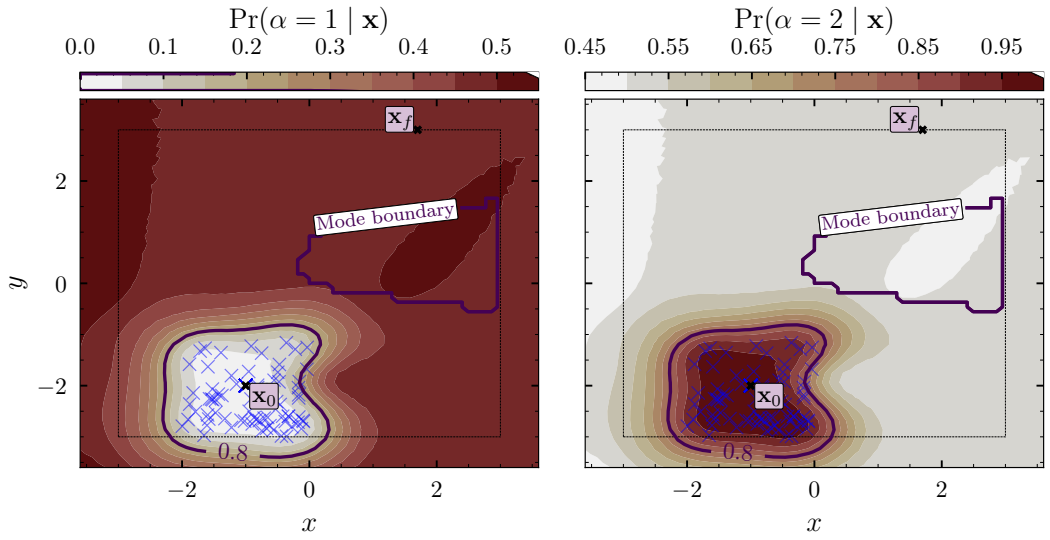
$$\mathbf{x}_T = \mathbf{x}_f, \tag{6.1f}$$

where the dynamics are given by Equation (2.1), the objective is given by Equation (2.2), and the mode remaining constraint in Equation (6.1c) is from the $\delta-$ mode remaining definition in Definition 4.1.1.

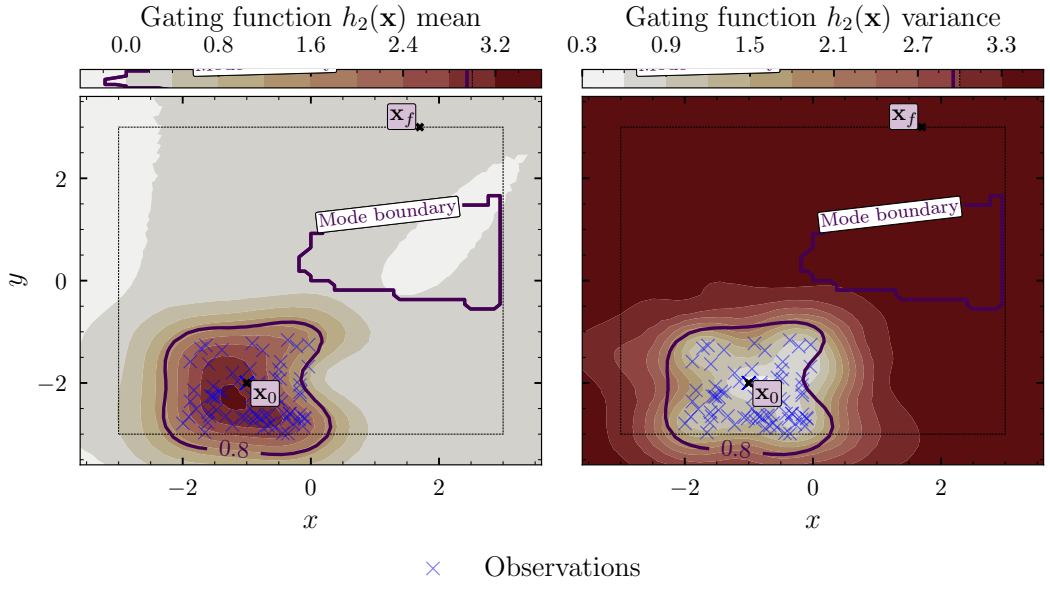
It is worth noting that the agent would not be able to explore the environment without relaxing the mode remaining constraint from Definition 2.1.1. Intuitively, the more the $\delta-$mode remaining constraint is relaxed, the more the controller π can explore. However, this will also increase the controller's chance of leaving the desired dynamics mode. Thus, the algorithm should expand the region that is known to belong to the desired dynamics mode towards the target state $\mathbf{x}_f$, without violating the $\delta-$ mode remaining constraint.

Initial mode remaining controller In robotics applications, an initial set of poor performing controllers can normally be obtained via simulation or domain knowledge. This work assumes access to an initial data set of state transitions $\mathcal{D}_0 = \{(\mathbf{x}_n, \mathbf{u}_n), \Delta\mathbf{x}_n\}_{n=1}^{N_0}$ collected from around the start state $\mathbf{x}_0$.

Assumption 6.1.1. *An initial region of the state space $\mathcal{X}_0 \subseteq \mathcal{X}_{k^*}$ is known to belong to the desired dynamics mode k^*. As such, a state transition data set can be collected $\mathcal{D}_0 = \{(\mathbf{x}_n, \mathbf{u}_n), \Delta\mathbf{x}_n\}_{n=1}^{N_0}$ such that it contains the start state $((\mathbf{x}_0, \cdot), \cdot) \in \mathcal{D}_0$.*



(a)



(b)

Figure 6.1: ***Initial gating network*** *Gating network posterior after training on the initial data set $\mathcal{D}_0$ shown by the blue crosses. The goal is to incrementally explore the environment until δ – mode remaining trajectories under the learned dynamics model, can be found from the start state $\mathbf{x}_0$, to the target state $\mathbf{x}_f$. The data set is overlayed on (a) each mode's mixing probability and (b) the posterior mean (left) and posterior variance (right) associated with the desired mode's gating function. The start state $\mathbf{x}_0$ and target state $\mathbf{x}_f$ are overlayed along with the mode boundary (purple line). All plots show a slice of the input space with constant zero controls.*

This data set is used to learn a predictive dynamics model p_θ which is locally accurate around the start state $\mathbf{x}_0$. Figure 6.1 shows the MoSVGPE's gating network posterior after training on the initial data set $\mathcal{D}_0$ for the quadcopter navigation problem in the illustrative example from Section 1.1. Figure 6.1a illustrates that the desired dynamics mode is $k^* = 2$ and Figure 6.1b shows that the GP posterior over the gating function is uncertain away from the initial data set $\mathcal{D}_0$, indicated by high posterior variance (red). This is further demonstrated by the probability mass function over the mode indicator variable in Figure 6.1a, tending to a uniform distribution, i.e. maximum entropy. Although this model can be used to learn an initial controller, it may not work outside of the initial state domain $\mathcal{X}_0$ and may not be able to find $\delta-$mode remaining trajectories to the target state $\mathbf{x}_f$, due to the model having high *epistemic uncertainty*.

6.2 Mode Optimisation

This section details our method for solving the mode remaining navigation problem by consolidating all of the work in this thesis. The method is named Mode Optimisation (ModeOpt). At its core, ModeOpt learns a single-step dynamics model p_θ using the MoSVGPE model from Chapter 3. Given this model, the mode remaining control methods in Chapter 4 can be used to find trajectories to the target state $\mathbf{x}_f$. The mode chance constraints from Equation (4.41) can then be used to check if these trajectories are $\delta-$mode remaining under the learned dynamics model p_θ. Initially, it will not be possible to find $\delta-$mode remaining trajectories under the learned dynamics model, due to high *epistemic uncertainty*. In this case, the agent must explore the environment, collect data and update its dynamics model. Eventually, the agent will reduce the model's *epistemic uncertainty* such that it can find $\delta-$mode remaining trajectories to the target state $\mathbf{x}_f$. Ideally, the exploration strategy will have some guarantee of remaining in the desired dynamics mode. Figure 6.2 illustrates this process in a flowchart.

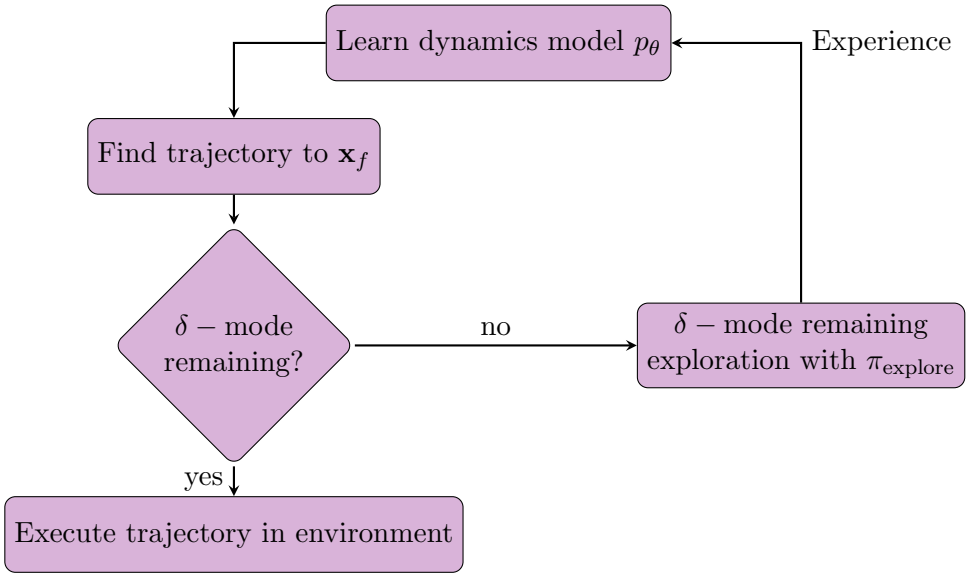


Figure 6.2: ***ModeOpt*** *Flowchart showing the sequence of steps and processes needed to control a system from an initial state $\mathbf{x}_0$, to a target state $\mathbf{x}_f$, with a $\delta-$mode remaining guarantee, when the underlying dynamics modes and how the system switches between them, are not fully known a priori.*

$\delta-$mode remaining exploration In order to implement such an algorithm, we require a method for exploring the environment with mode remaining guarantees. That is, a controller that will explore the environment whilst ensuring the controlled system remains in the desired dynamics mode with high probability, i.e. satisfy Equation (4.2).

6.2.1 Mode Remaining Exploration

The performance of ModeOpt depends on its ability to explore the environment. The exploration strategy is primarily interested in exploring a single desired dynamics mode whilst avoiding entering any of the other modes. This is a challenging problem because the agent must observe regions outside of the desired dynamics mode in order to learn that a particular region does not belong to the desired mode. To the best of our knowledge, there is no previous work addressing the exploration of a single dynamics mode in multimodal dynamical systems.

$\delta-$**mode remaining exploration** The exploration strategy presented here is primarily interested in exploring regions of the gating network that are uncertain. It relieves the myopia of active learning by considering trajectory optimisation with an entropy-based strategy over a finite horizon. Further to this, it principally ensures solutions are $\delta-$mode remaining via the mode chance constraints in Equation (4.41). The objective combines entropy-based exploration of the gating network with the main goal of navigating to the target state $\mathbf{x}_f$. The exploration strategy is given by,

$$\max_{\pi\in\Pi} \underbrace{\mathcal{H}[h_{k^*}(\bar{\mathbf{x}}) \mid \bar{\mathbf{x}}, \mathcal{D}_{0:i-1}]}_{\text{joint gating entropy}} \tag{6.2a}$$

$$+ \sum_{t=1}^{T-1} \mathbb{E}\left[\underbrace{-(\mathbf{x}_t - \mathbf{x}_f)^T \mathbf{Q}(\mathbf{x}_t - \mathbf{x}_f)}_{\text{state difference}} - \underbrace{\mathbf{u}_t^T \mathbf{R}\mathbf{u}_t}_{\text{control cost}}\right] \tag{6.2b}$$

$$\text{s.t. } \mathbf{x}_{t+1} \sim p_\theta(\mathbf{x}_{t+1} \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \bar{\boldsymbol{\alpha}}_{0:t} = k^*) \qquad \forall t \in \{0, \ldots, T-1\} \tag{6.2c}$$

$$\Pr(\alpha_t = k^* \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \boldsymbol{\alpha}_{0:t-1} = k^*) \geq 1 - \delta \qquad \forall t \in \{0, \ldots, T\} \tag{6.2d}$$

$$\mathbf{u}_t = \pi(\mathbf{x}_t, t) \in \mathcal{U} \qquad \forall t \in \{0, \ldots, T-1\} \tag{6.2e}$$

$$\mathbf{x}_0 = \mathbf{x}_0, \tag{6.2f}$$

where $p_\theta(\mathbf{x}_{t+1} \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \bar{\boldsymbol{\alpha}}_{0:t} = k^*)$ is the multi-step dynamics prediction calculated by recursively propagating uncertainty through the desired mode's dynamics GP using the moment matching approximation. See Equation (4.40) for more details. $\mathbf{Q}$ and $\mathbf{R}$ are user-defined, real, symmetric, positive semi-definite and positive definite matrices respectively. Intuitively, the first (joint gating entropy) term seeks to find trajectories that explore regions of the gating network with high *epistemic uncertainty* and the second (state difference) term targets exploration towards the target state $\mathbf{x}_f$. The joint gating entropy prevents trajectories from collapsing onto a single location of high entropy as it favours trajectories spreading over the state space in order to maximise the entropy of the entire trajectory. The state difference

term favours trajectories whose centre of mass is closer to the target state $\mathbf{x}_f$. The control cost term regularises the control trajectory.

Chance constraints To ensure that the controlled system remains in the desired mode with high probability, i.e. trajectories are $\delta -$ mode remaining, this method deploys the chance constraints from Equation (4.41). In practice, the constrained optimisation in Equation (6.2) fails if the initial trajectory does not satisfy the mode chance constraints. As such, this work deploys a simple strategy to ensure the initial trajectory satisfies the constraints. We sample a "fake" target state from the initial data set $\mathcal{D}_0$ and optimise the initial trajectory using the cost function in Equation (4.65). This finds a straight line trajectory between the start state $\mathbf{x}_0$ and the "fake" target state which is in the initial state domain. Experiments show that this procedure finds trajectories where all of the time steps are in the desired mode with high probability.

Gating network entropy To calculate the joint gating entropy term $\mathcal{H}[h_{k^*}(\bar{\mathbf{x}}) \mid \bar{\mathbf{x}}, \mathcal{D}_{0:i-1}]$ the state trajectory $\bar{\mathbf{x}}$ is first obtained by cascading single-step predictions through the desired mode's dynamics GP using the moment matching approximation from Kuss, 2006. The key idea is then to assume that the gating function values along the trajectory $h_{k^*}(\bar{\mathbf{x}})$ are jointly Gaussian according to the GP over the desired mode's gating function. The joint distribution over the gating function values $h_{k^*}(\bar{\mathbf{x}})$ is then given by,

$$p(h_{k^*}(\bar{\mathbf{x}}) \mid \bar{\mathbf{x}}, \mathcal{D}_{0:i-1}) \approx q(h_{k^*}(\bar{\mathbf{x}})) = \mathcal{N}\big(h_{k^*}(\bar{\mathbf{x}}) \mid \boldsymbol{\mu}_{k^*}(\bar{\mathbf{x}}), \boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{x}}, \bar{\mathbf{x}})\big) \tag{6.3}$$

where $\boldsymbol{\mu}_{k^*}(\cdot)$ and $\boldsymbol{\Sigma}^2_{k^*}(\cdot, \cdot)$ are sparse GP mean and covariance functions, given by,

$$\boldsymbol{\mu}_{k^*}(\bar{\mathbf{x}}) = \hat{k}_{k^*}(\bar{\mathbf{x}}, \boldsymbol{\xi})\hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1}\hat{\mathbf{m}}_{k^*} \tag{6.4}$$

$$\boldsymbol{\Sigma}^2_{k^*}(\bar{\mathbf{x}}, \bar{\mathbf{x}}) = \hat{k}_{k^*}(\bar{\mathbf{x}}, \bar{\mathbf{x}}) + \hat{k}_{k^*}(\bar{\mathbf{x}}, \boldsymbol{\xi})\hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1}\Big(\hat{\mathbf{S}}_{k^*} - \hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})\Big)\hat{k}_{k^*}(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1}\hat{k}_{k^*}(\boldsymbol{\xi}, \bar{\mathbf{x}}), \tag{6.5}$$

Algorithm 2 ModeOpt

Require: Start state $\mathbf{x}_0$, target state $\mathbf{x}_f$, desired dynamics mode k^*, initial data set $\mathcal{D}_0$, explorative controller π_{explore}, mode remaining controller π_{mode}, dynamics model p_θ

for $i = 0, 1, \ldots$ **do**
  Train model p_θ on $\mathcal{D}_i$ using ELBO from Equation (3.29)
  Optimise mode remaining controller π_{mode} using learned model p_θ
  if $\mathbf{x}_T = \mathbf{x}_f$ and π_{mode} is $\delta-$mode remaining **then**
    Execute π_{mode} in environment
    break
  end if
  Optimise explorative controller π_{explore} using learned model p_θ
  Collect environment data set $\mathcal{D}_{i+1}$ using π_{explore}; add to data set$\{\mathcal{D}_{0:i+1} = \mathcal{D}_{i+1} \cup \mathcal{D}_{0:i}\}$
end for

where $\hat{k}_{k^*}$ and $\boldsymbol{\xi}$ are the kernel and inducing inputs associated with the desired mode's gating function respectively. This sparse approximation arises because the MoSVGPE model uses sparse GPs and approximates the posterior with,

$$q(h_{k^*}(\bar{\mathbf{x}})) = \int p(h_{k^*}(\bar{\mathbf{x}}) \mid h_{k^*}(\boldsymbol{\xi}))q(h_{k^*}(\boldsymbol{\xi}))\mathrm{d}h_{k^*}(\boldsymbol{\xi}), \tag{6.6}$$

where $q(h_{k^*}(\boldsymbol{\xi})) = \mathcal{N}\Big(h_{k^*}(\boldsymbol{\xi} \mid \hat{\mathbf{m}}_{k^*}, \hat{\mathbf{S}}_{k^*}\Big)$.

6.2.2 Mode Remaining Model-based Reinforcement Learning

This section details how this exploration strategy is embedded into a MBRL loop to solve the mode remaining navigation problem in Equation (6.1). The algorithm is named ModeOpt and is detailed in Algorithm 2. The algorithm is initialised with a start state $\mathbf{x}_0$, a target state $\mathbf{x}_f$, a desired dynamics mode k^*, a data set of state transitions from the desired dynamics mode $\mathcal{D}_0$ and a calibrated dynamics model p_θ.

Dynamics model p_θ ModeOpt learns a factorised representation of the underlying dynamics modes using the MoSVGPE model from Chapter 3. Importantly, it learns a single-step dynamics model p_θ, given by,

$$\mathbf{x}_{t+1} \sim p_\theta(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t), \tag{6.7}$$

which also infers valuable information regarding how the system switches between the modes.

Mode remaining controller $\pi_{\mathbf{mode}}$ The mode remaining control methods from Chapter 4 are used to construct a mode remaining controller π_{mode},

$$\mathbf{u}_t = \pi_{\text{mode}}(t) \quad \forall t \in \{0, \dots, T-1\}. \tag{6.8}$$

It uses the learned dynamics model p_θ to find trajectories to the target state $\mathbf{x}_f$. The mode chance constraints from Equation (4.41) are then used to check if trajectories are $\delta-$mode remaining.

Explorative controller $\pi_{\mathbf{explore}}$ When $\delta-$mode remaining trajectories to the target state $\mathbf{x}_f$ cannot be found, ModeOpt relies on an explorative controller π_{explore} to explore the environment. This work uses the exploration strategy from Section 6.2.1 to construct such an explorative controller,

$$\mathbf{u}_t = \pi_{\text{explore}}(t) \quad \forall t \in \{0, \dots, T-1\}. \tag{6.9}$$

The goal of this controller is to explore the environment whilst remaining in the desired dynamics mode. It is worth noting that this is an open-loop trajectory optimisation algorithm. Extending the method in Section 6.2.1 to feedback controllers is left for future work.

6.3 Preliminary Results

This section presents initial results solving the illustrative quadcopter navigation problem in Section 1.1 using the exploration strategy from Section 6.2.1. In particular, it shows results using the simulated Environment 1 from Section 5.2. See Section 5.2 for more details on the environment and the simulator set up.

Figure 6.1 shows the MoSVGPE's gating network posterior after training on the initial data set $\mathcal{D}_0$. The start state $\mathbf{x}_0$, the target state $\mathbf{x}_f$, the initial data set $\mathcal{D}_0$ (blue crosses) and the mode boundary, are overlayed to help visualise the mode remaining navigation problem. The turbulent dynamics mode is the region within the mode boundary and the desired dynamics mode is everywhere else. To further aid visualisation, the mode chance constraints contour is marked on all plots (purple line), i.e. $\Pr(\alpha = k^* \mid \mathbf{x}, \mathcal{D}_{0:i}) = 1 - \delta$. Intuitively, ModeOpt seeks to expand the $1 - \delta$ contour until the target state $\mathbf{x}_f$ lies within the contour.

6.3.1 Experiment Configuration

This section details how the experiments were configured.

Initial data set $\mathcal{D}_0$ The initial data set was collected by simulating 15 random trajectories from the start state $\mathbf{x}_0$ and terminating them when they left the initial state domain $\mathcal{X}_0$. This resulted in an initial data set of 118 state transitions, which is visualised as the blue crosses in Figure 6.1.

Model learning Following the experiments in Chapter 5 this section instantiated the MoSVGPE dynamics model with $K = 2$ experts, one to represent each of the dynamics modes. Each mode's dynamics GP used a Squared Exponential kernel with ARD and a constant mean function. The gating network used a single gating function with a Squared Exponential kernel with ARD and a zero mean function. At each ModeOpt iteration, an early stopping callback was used to terminate the dynamics model training. The early stopping callback used a min delta of 0 and

a patience of 1200. This meant that training terminated after 1200 epochs of no improvement.

Explorative controller The explorative controller π_{explore} was initialised with a horizon of $T = 15$. At each ModeOpt iteration, the previous solution was used as the initial solution for the trajectory optimiser. In all experiments, ModeOpt was configured with $\delta = 0.2$, which corresponds to mode chance constraints with a satisfaction probability of 0.8. That is, the mode chance constraints were given by,

$$\Pr(\alpha_t = k^* \mid \mathbf{x}_0, \mathbf{u}_{0:t}, \boldsymbol{\alpha}_{0:t-1} = k^*) \geq 0.8 \quad \forall \in \{0, \dots, T-1\}. \tag{6.10}$$

6.3.2 Comparison of Exploration Terms

This section evaluates the different terms in the explorative controller's objective from Equation (6.2). In particular, it motivates why the entropy-based term was combined with the state difference term. It then shows why the joint gating entropy over a trajectory was used, instead of summing the marginal entropy at each time step, as is done in Buisson-Fenet et al., 2020.

State difference term A simple exploration strategy is to favour trajectories whose centre of mass is closer to the target state. This corresponds to solving the constrained optimisation in Equation (6.2) with only the state difference term. However, using only the state difference term with the mode chance constraints leads to the optimisation getting stuck in a local optimum and never exploring to the target state. This is shown in Figure 6.3, which shows the final iteration of the optimisation where the trajectory is stuck at the mode boundary. If the strategy searched more to the left it would be able to expand the mode chance constraints and explore around the mode boundary. However, the mode chance constraints have induced a local optimum that prevented ModeOpt from exploring in this direction. A strategy which favours searching regions of the state space which have not previously been observed

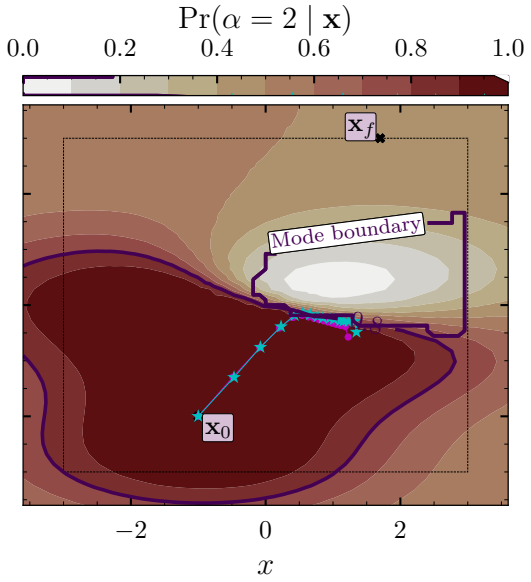
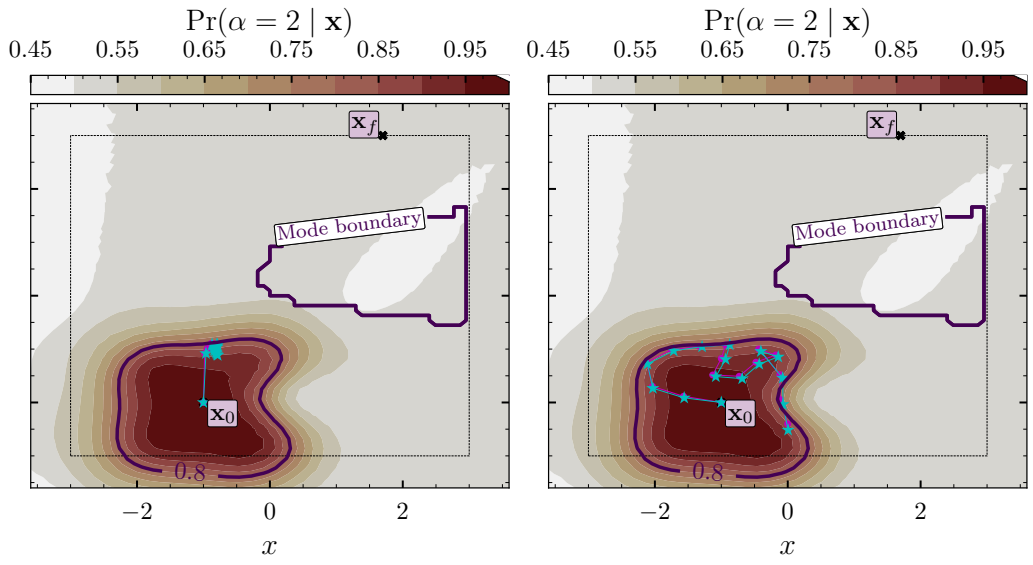


Figure 6.3: ***State difference only*** *Results when solving the constrained optimisation in Equation (6.2) with only the state difference term, i.e.* $J(f, \pi) = \sum_{t=1}^{T-1} \mathbb{E}\big[-(\mathbf{x}_t - \mathbf{x}_f)^T \mathbf{Q} (\mathbf{x}_t - \mathbf{x}_f)\big]$*. It shows the iteration where the optimisation gets stuck and no longer explores. The optimised controls are rolled out in the desired mode's GP dynamics (magenta) and in the environment (cyan) and are overlayed on the desired mode's mixing probability.*

could likely avoid the local optima in Figure 6.3. This intuition motivated adding the gating entropy term in Equation (6.2), which favours exploration in regions of high *epistemic uncertainty*.

Joint vs factorised entropy Before combining the state difference and entropy terms into a single objective, the impact of different gating entropy objectives is evaluated. In particular, the joint gating entropy term in Equation (6.2), given by,

$$J_{\text{joint}}(f, \pi) = H[h_{k^*}(\bar{\mathbf{x}}) \mid \bar{\mathbf{x}}, \mathcal{D}_{0:i-1}], \tag{6.11}$$



(a) Factorised entropy Equation (6.12) *(b) Joint entropy Equation (6.11)*

Figure 6.4: ***Comparison of factorised/joint entropy objectives*** *Comparison of solving the constrained optimisation in Equation (6.2) with (a) the sum of marginal entropies at each time step (factorised entropy) and (b) the full joint entropy over a trajectory. The trajectory found with the factorised entropy collapses onto a single location of high entropy. The optimised controls are rolled out in the desired mode's GP dynamics (magenta) and in the environment (cyan) and are overlayed on each mode's mixing probability.*

is compared with the sum of marginal entropies over the trajectory, which is referred to as the factorised entropy and given by,

$$J_{\text{fact}}(f, \pi) = \sum_{t=0}^{T} H[h_{k^*}(\mathbf{x}_t) \mid \mathbf{x}_t, \mathcal{D}_{0:i-1}]. \tag{6.12}$$

Figure 6.4 shows the trajectories found at the first ModeOpt iteration when using these two entropy objectives. The factorised entropy objective, shown in Figure 6.4a, has collapsed the entire trajectory onto a single state of high entropy. This is an undesirable behaviour because it does not maximise the information gain along the entire trajectory. In contrast, the result in Figure 6.4b considers the joint entropy over the entire trajectory. The trajectory has spread along the $1 - \delta$ contour, which corresponds to the region with the highest gating entropy. These results confirm that it is important to consider the information gain over the entire trajectory.

6.3.3 Exploration in Environment 1

This section presents preliminary results of ModeOpt's exploration strategy. The results show the strategy successfully exploring the simulated Environment 1 from Section 5.2. The results presented here show how the desired mode's mixing probability and the gating function's variance change during each iteration. The exploration strategy is visualised using a grid of figures where each row corresponds to the data collection process corresponding to a particular iteration i. For example, Figure 6.5 shows how the desired mode's mixing probability changes at the start of the exploration phase and Figure 6.6 shows how the gating function's variance changes.

1. The first (left) column shows the data set $\mathcal{D}_{0:i}$ collected at the previous iterations overlayed on the gating network posterior **after** training on it.
2. The second (middle) column overlays the trajectory found by the explorative controller π_{explore} on the gating network posterior **before** training on the data collected by the explorative controller.
3. The third (right) column shows the updated data set $\mathcal{D}_{0:i+1}$ after collecting data with the explorative controller, overlayed on the gating network **before** training on it.

The row below then shows the gating network posterior **after** training on the updated data set $\mathcal{D}_{0:i+1}$ from the previous iteration. Together Figures 6.5 and 6.6 show how the model's *epistemic uncertainty* reduces after collecting and training on new data, and how this leads to the mode chance constraints expanding at each iteration.

Initial model The top left figures in Figures 6.5 and 6.6 show the initial data set $\mathcal{D}_0$ (blue crosses) overlayed on the gating network after training on it. Figure 6.5 shows that the probability of being in the desired dynamics mode $k^* = 2$ is high (red) around the initial data set. It tends to 0.5 away from the data, which corresponds to maximum entropy for a Categorical distribution. Figure 6.6 shows that

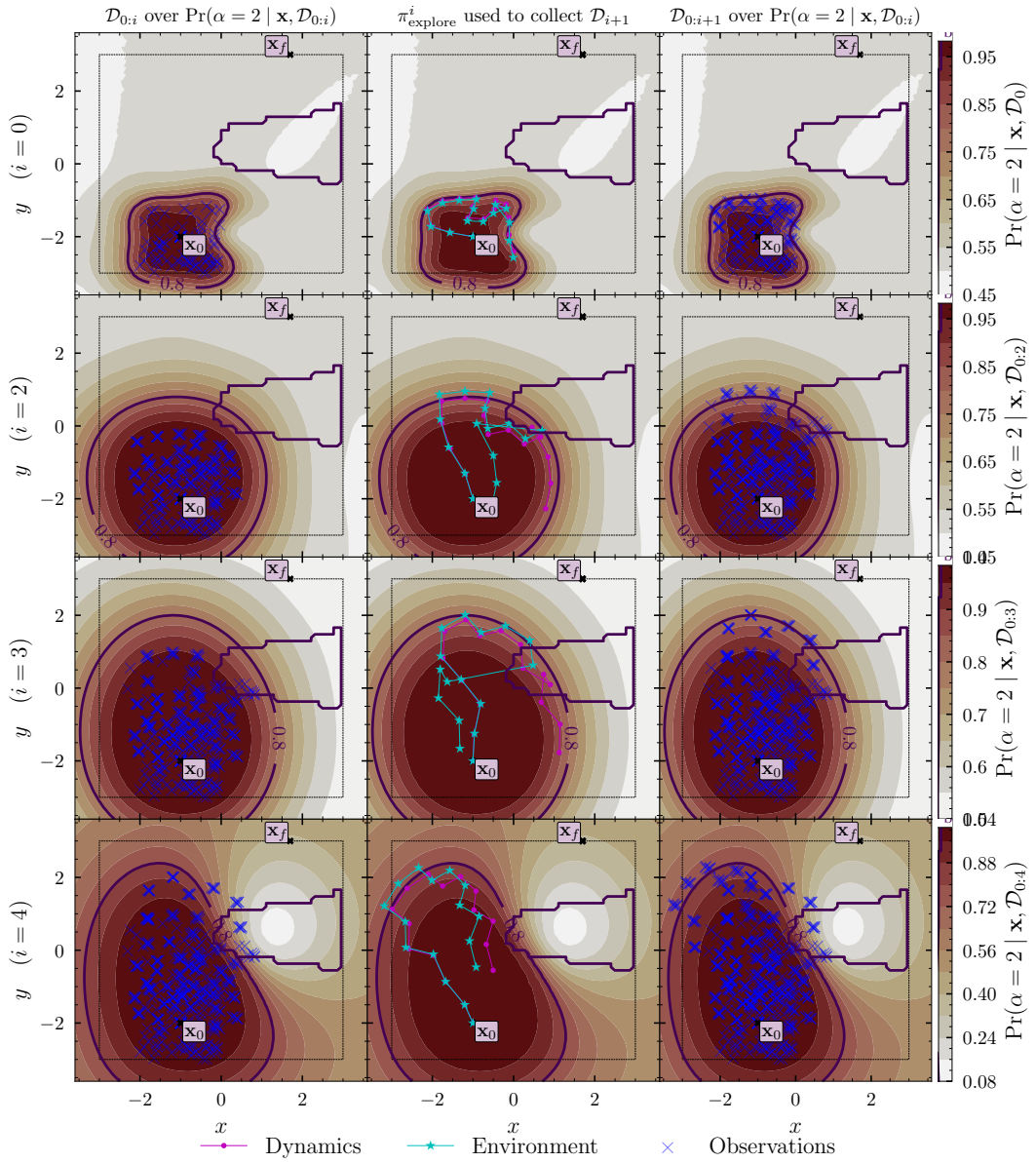


Figure 6.5: ***ModeOpt iterations*** $\mathbf{i = 0, 2, 3, 4}$ ***over mode probability*** *Visualisation of ModeOpt at the start of it's exploration phase in Environment 1 overlayed on the desired mode's mixing probability. It shows how the explorable region – indicated by the boundary of the mode chance constraints (purple 0.8 contour) – expands after training on data collected using the explorative controller $\pi_{explore}$. Reading left to right shows the data collection process whilst top to bottom shows the dynamics model updating. Each row shows the data collection process at a given iteration i which has used the dynamics model* ***after*** *training on the previous data set $\mathcal{D}_{0:i}$ to collect a new data set $\mathcal{D}_{i+1}$. The data collection process is overlayed on the desired mode's mixing probability* ***after*** *training on the data set collected at the previous iteration $\Pr(\alpha = k^* \mid \mathbf{x}, \mathcal{D}_{0:i})$. The left plots show the data set collected at the previous iteration $\mathcal{D}_{0:i}$ (blue crosses). The middle plots show the trajectory found by the explorative controller rolled out in the desired mode's GP dynamics (magenta) and in the environment (cyan). The right plots show the updated data set $\mathcal{D}_{0:i+1}$ (blue crosses) after collecting data with the explorative controller.*

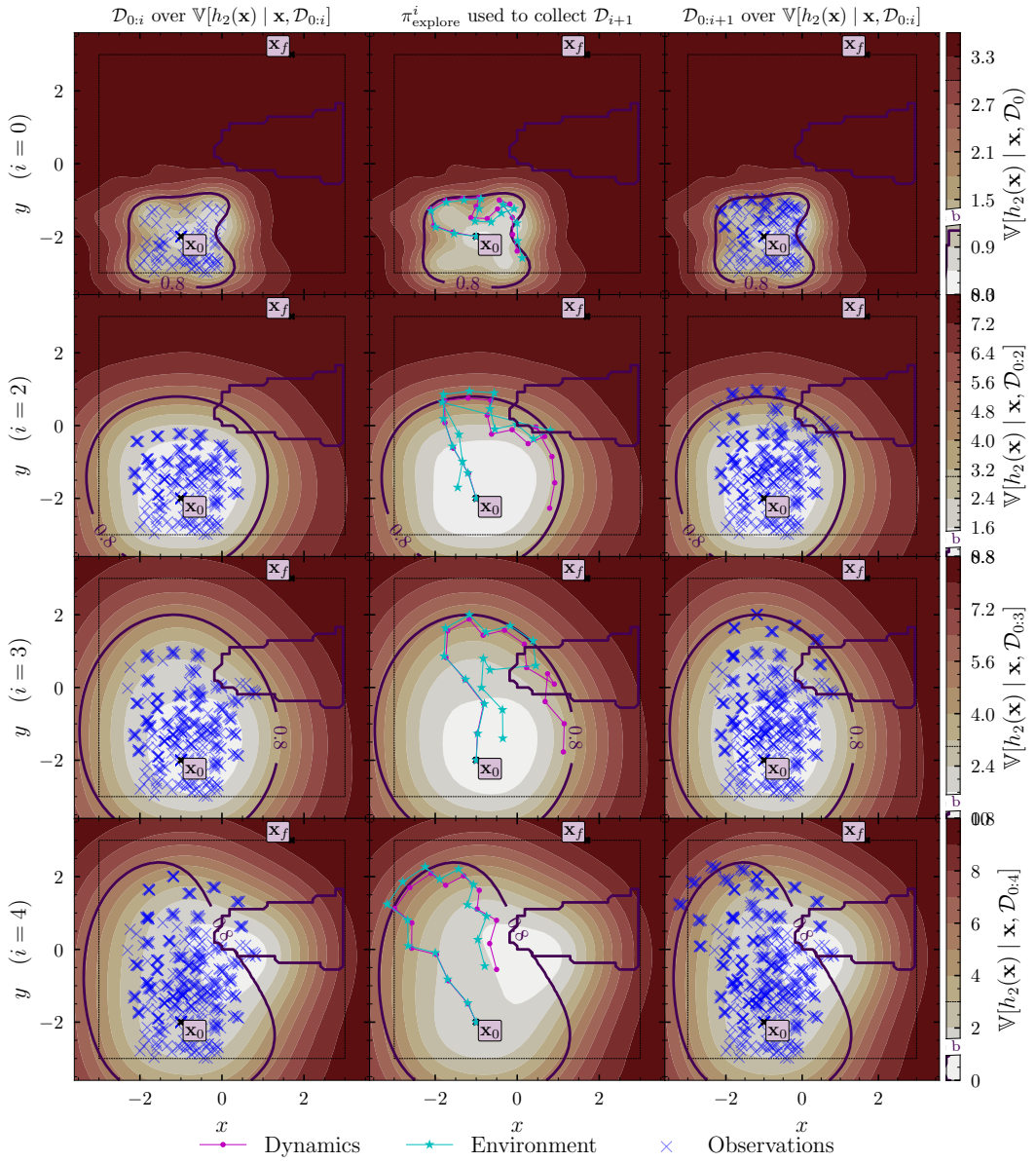


Figure 6.6: ***ModeOpt iterations*** $\mathbf{i} = \mathbf{0}, \mathbf{2}, \mathbf{3}, \mathbf{4}$ ***over gating function variance*** *Visualisation of ModeOpt at the start of it's exploration phase in Environment 1 overlayed on the gating function's posterior variance. It shows the explorable region – indicated by the boundary of the mode chance constraints (purple 0.8 contour) – expanding after training on data collected using the explorative controller $\pi_{explore}$. Reading left to right shows the data collection process whilst top to bottom shows the dynamics model updating. Each row shows the data collection process at iteration i which uses the dynamics model* ***after*** *training on the previous data set $\mathcal{D}_{0:i}$ to collect a new data set $\mathcal{D}_{i+1}$. The data collection process is overlayed on the gating function's variance* ***after*** *training on the previous data set $\mathbb{V}[h_{k^*}(\mathbf{x}) \mid \mathbf{x}, \mathcal{D}_{0:i}]$. The left plots show the data set collected at the previous iteration $\mathcal{D}_{0:i}$ (blue crosses). The middle plots show the trajectory found by the explorative controller rolled out in the desired mode's GP dynamics (magenta) and in the environment (cyan). The right plots show the updated data set $\mathcal{D}_{0:i+1}$ (blue crosses) after collecting data with the explorative controller.*

the gating function's posterior variance is low around the initial data set and high everywhere else. Intuitively, ModeOpt seeks to expand the explorable region so that $\delta -$ mode remaining trajectories to the target state can be found. As such, it seeks to reduce the model's *epistemic uncertainty* by decreasing the gating function's variance. In turn, this will increase the region that the model believes belongs to the desired dynamics mode. This is visualised by the chance constraint contour $\Pr(\alpha = k^* \mid \mathbf{x}, \mathcal{D}_{0:i}) = 0.8$ expanding as ModeOpt collects more data and trains on it.

Iteration i = 0 The top row of Figures 6.5 and 6.6 visualise the initial iteration $i = 0$ of ModeOpt. Given the dynamics model after training on the initial data set $\mathcal{D}_0$, the explorative controller π_{explore} found the trajectory in the top middle plot. The trajectory explores towards the target state $\mathbf{x}_f$, which is the goal of the state difference term in Equation (6.2). The top middle plot in Figure 6.6 shows that the trajectory has also spread over regions of the desired mode's gating function with high posterior variance. This is the goal of the joint gating entropy term in Equation (6.2). The contour plots in the second row show how the gating network changes after training on $\mathcal{D}_{0:1}$. The gating function's posterior variance has decreased around the new observations. The purple lines representing the $\Pr(\alpha = k^* \mid \mathbf{x}, \mathcal{D}_0) = 0.8$ contour in the top row, and the $\Pr(\alpha = k^* \mid \mathbf{x}, \mathcal{D}_{0:1}) = 0.8$ contour in the second row, show how the mode chance constraints expand after training on $\mathcal{D}_{0:1}$. The region between the contours represents the newly explorable region. This shows that reducing the model's *epistemic uncertainty* leads to the explorable region expanding under the mode chance constraints. Further to this, it indicates that in this environment, setting $\delta = 0.2$ enables exploration outside of the initial state domain $\mathcal{X}_0$.

Iteration i = 2 The second row of plots in Figure 6.5 shows the first iteration where the $\delta -$ mode remaining chance constraints have expanded the explorable domain such that it intersects with the undesired dynamics mode. This is indicated by the purple 0.8 contour intersecting with the purple line labelled mode boundary. As a

result, the trajectory found by the explorative controller left the desired dynamics mode and was subject to the turbulent dynamics mode. This is shown by the trajectory crossing the mode boundary and also by the environment trajectory (cyan) deviating from the dynamics trajectory (magenta) due to the high drift associated with the turbulent dynamics mode. It is worth noting that ModeOpt can never learn where a mode boundary is without observing state transitions from outside of the desired dynamics mode. However, ModeOpt was not able to learn the mode boundary from this single trajectory.

Iteration $\mathbf{i = 3}$ The third row shows another iteration where the dynamics model did not learn the mode boundary and as a result, found an explorative trajectory which left the desired dynamics mode. Although not desirable, collecting this data was necessary for inferring the mode boundary.

Iteration $\mathbf{i = 4}$ The fourth row shows the next iteration which shows that given these new observations the MoSVGPE's gating network is able to infer that the state transitions belong to another dynamics mode. As a result, the explorative trajectory did not leave the desired dynamics mode. That is, the mode chance constraints successfully restricted the domain such that the trajectory did not leave the desired dynamics mode. Further to this, with $\delta = 0.2$ the trajectory optimiser was able to explore around the mode boundary without leaving the desired dynamics mode. This confirms that the exploration strategy in Section 6.2.1 is capable of exploring subject to the constrained domain.

Leaving the desired dynamics mode The interplay between the explorative objective and the mode chance constraints was effective at preventing the explorative controller from leaving the desired dynamics mode during training. Only two iterations of ModeOpt left the desired dynamics mode during training and arguably this is necessary in order to learn the location of the mode boundary.

Final iteration Figures 6.7 and 6.8 show the later iterations of the exploration phase. They show ModeOpt gradually exploring the domain by expanding the mode chance constraints. The bottom row shows the final iteration of the exploration

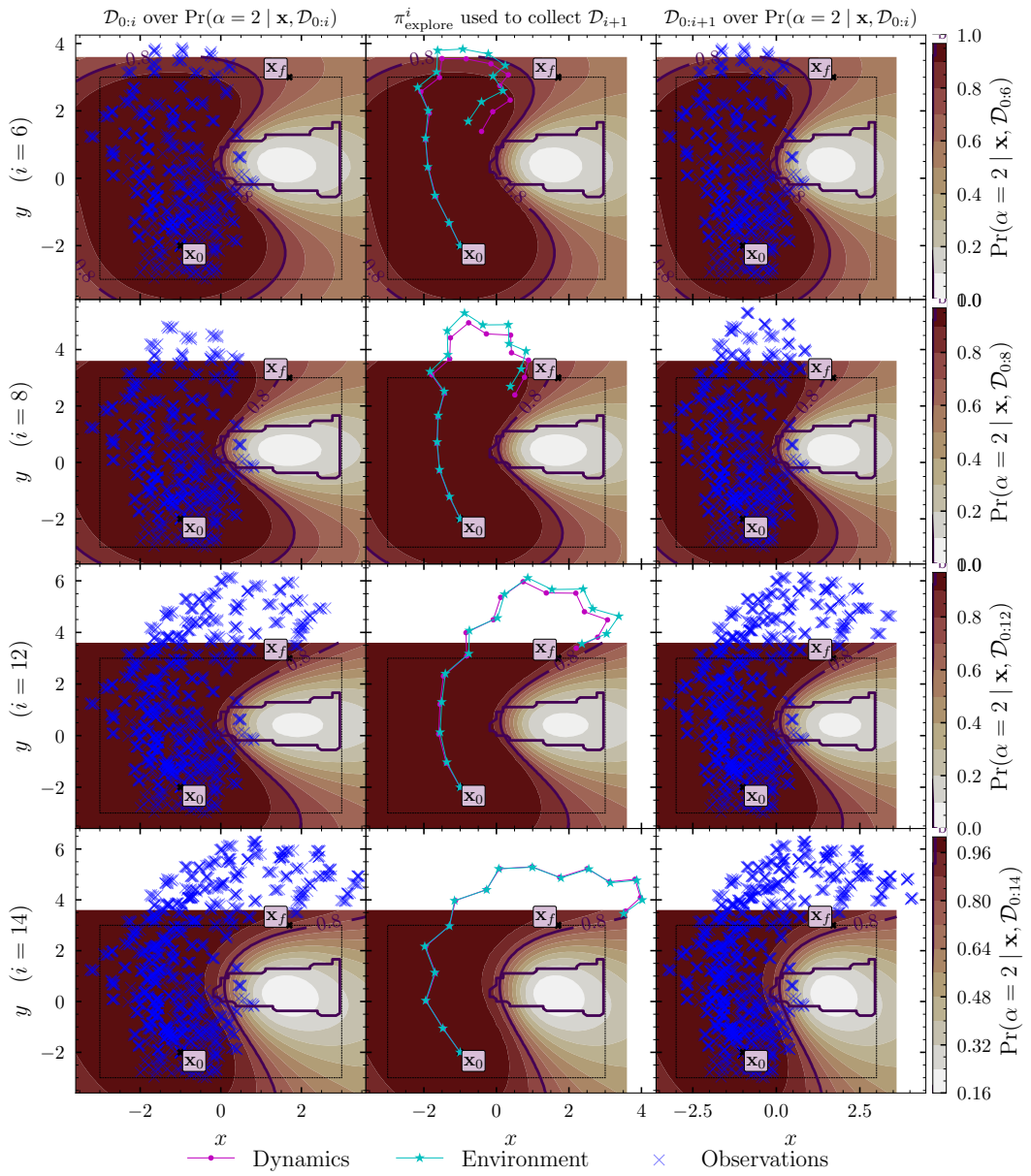


Figure 6.7: ***ModeOpt iterations*** $\mathbf{i = 6, 8, 12, 14}$ ***over mode probability*** *Visualisation of ModeOpt at the* ***end*** *of it's exploration phase in Environment 1 overlayed on the desired mode's mixing probability. It shows how the explorable region – indicated by the boundary of the mode chance constraints (purple 0.8 contour) – expands after training on data collected using the explorative controller $\pi_{explore}$. Reading left to right shows the data collection process whilst top to bottom shows the dynamics model updating. Each row shows the data collection process at a given iteration i which has used the dynamics model* ***after*** *training on the previous data set $\mathcal{D}_{0:i}$ to collect a new data set $\mathcal{D}_{i+1}$. The data collection process is overlayed on the desired mode's mixing probability* ***after*** *training on the data set collected at the previous iteration $\Pr(\alpha = k^* \mid \mathbf{x}, \mathcal{D}_{0:i})$. The left plots show the data set collected at the previous iteration $\mathcal{D}_{0:i}$ (blue crosses). The middle plots show the trajectory found by the explorative controller rolled out in the desired mode's GP dynamics (magenta) and in the environment (cyan). The right plots show the updated data set $\mathcal{D}_{0:i+1}$ (blue crosses) after collecting data with the explorative controller.*

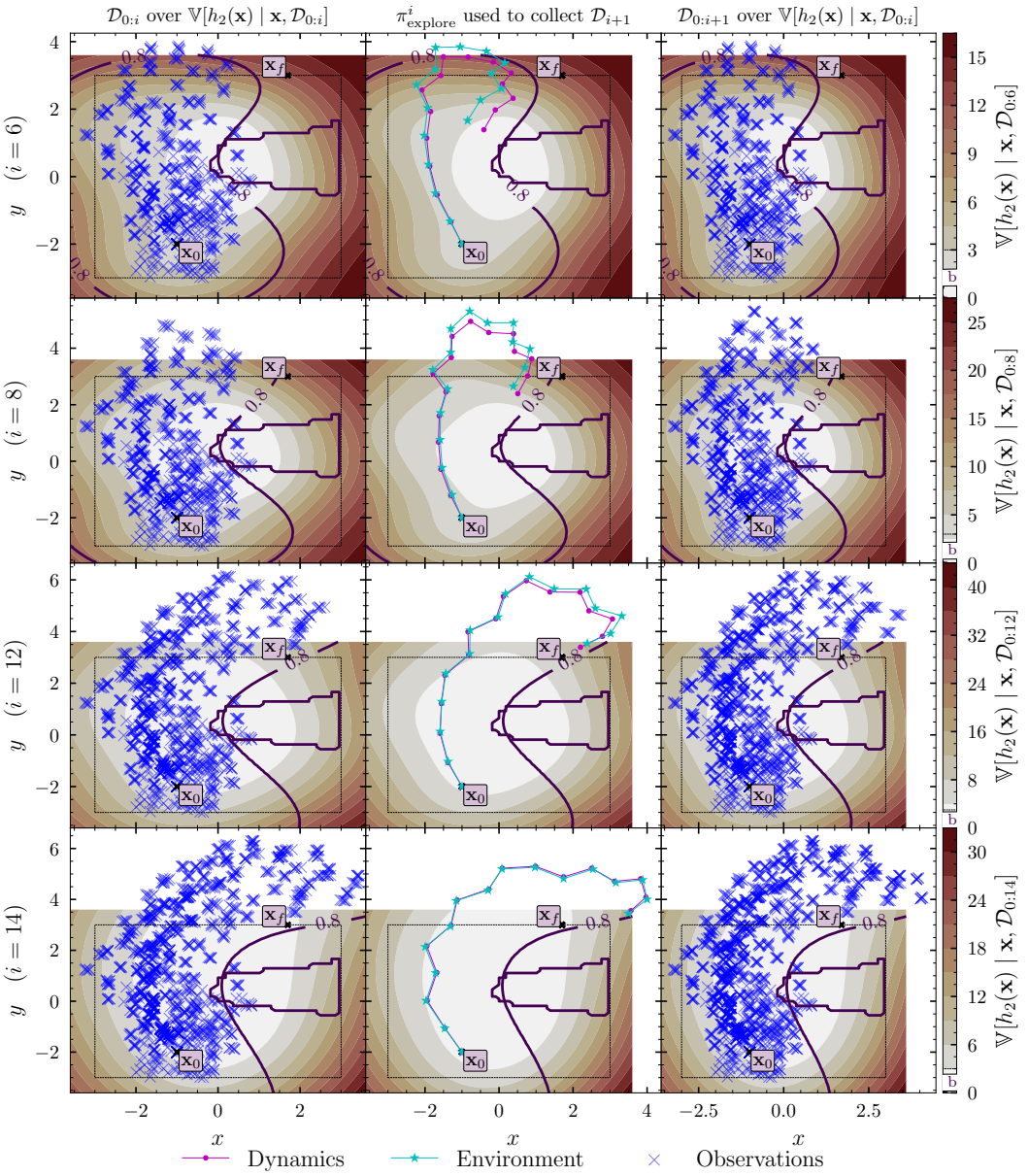


Figure 6.8: ***ModeOpt iterations*** $\mathbf{i = 6, 8, 12, 14}$ ***over gating function variance*** *Visualisation of ModeOpt at the* ***end*** *of it's exploration phase in Environment 1 overlayed on the gating function's posterior variance. It shows the explorable region – indicated by the boundary of the mode chance constraints (purple 0.8 contour) – expanding after training on data collected using the explorative controller $\pi_{explore}$. Reading left to right shows the data collection process whilst top to bottom shows the dynamics model updating. Each row shows the data collection process at iteration i which uses the dynamics model* ***after*** *training on the previous data set $\mathcal{D}_{0:i}$ to collect a new data set $\mathcal{D}_{i+1}$. The data collection process is overlayed on the gating function's variance* ***after*** *training on the previous data set $\mathbb{V}[h_{k^*}(\mathbf{x}) \mid \mathbf{x}, \mathcal{D}_{0:i}]$. The left plots show the data set collected at the previous iteration $\mathcal{D}_{0:i}$ (blue crosses). The middle plots show the trajectory found by the explorative controller rolled out in the desired mode's GP dynamics (magenta) and in the environment (cyan). The right plots show the updated data set $\mathcal{D}_{0:i+1}$ (blue crosses) after collecting data with the explorative controller.*

phase where the mode chance constraints have expanded such that the target state $\mathbf{x}_f$ lies within the explorable domain. This result confirms that the constrained exploration strategy is capable of exploring to the target state $\mathbf{x}_f$ with $\delta = 0.2$. However, it is worth noting that the mode chance constraints intersect the mode boundary at the final iteration. As such $\delta-$mode remaining trajectories found with the mode remaining controller π_{mode} may leave the desired dynamics mode. In this case, ModeOpt requires more observations around the mode boundary to learn the true position of the mode boundary. This issue arises because the constraints are latent and are being inferred from data.

Exploration The results in Figures 6.5 to 6.8 suggest that balancing the state difference and the joint gating entropy terms in Equation (6.2) results in exploration to the target state $\mathbf{x}_f$ without getting stuck in a local minimum. Intuitively, this objective favours maximum entropy trajectories whose centre of mass is closest to the target state $\mathbf{x}_f$. It is worth noting that not all settings of the cost matrices, $\mathbf{Q}$ and $\mathbf{R}$, resulted in ModeOpt converging in the experiments. As mentioned previously, the performance of ModeOpt's exploration is dependent on the interplay between the entropy term, the state difference term and the mode chance constraints. As such, the convergence of ModeOpt likely depends on the cost matrices, $\mathbf{Q}$, $\mathbf{R}$ and the mode satisfaction probability given by our choice of δ. Exploration can likely be guaranteed by relaxing δ, however, relaxing δ too far corresponds to removing the mode chance constraints. Nevertheless, the results shown here are an initial step showing that ModeOpt can work in practice. However, further analysis is left for future work.

6.4 Discussion & Future Work

This section discusses ModeOpt and proposes some directions for future work.

Further experiments First of all, it should be noted that the work presented in this chapter is a first step at consolidating the work in this thesis to solve the mode remaining navigation problem in Equation (6.1). As such, more experiments are required to fully test ModeOpt. For example, further experiments are required to validate the exploration strategy in environments with more than two dynamics modes and on real-world systems.

Exploration guarantees ModeOpt side-steps the exploration-exploitation trade-off which is common in MBRL. It does so by separating the exploration into the explorative controller π_{explore} and the exploitation into the mode remaining controller π_{mode}. The mode chance constraints are then used to see if the exploitative mode controller π_{mode} can find a $\delta-$mode remaining trajectory to the target state. If it cannot, it falls back on the explorative controller π_{explore} to find informative trajectories, collect data and reduce the model's *epistemic uncertainty*. Although this approach principally side-steps the exploration-exploitation trade-off, it does not have any exploration guarantees. That is, given enough time, it is not guaranteed to have explored enough such that $\delta-$mode remaining trajectories to the target state can be found. Further to this, some experiments showed that the constrained optimisation can get stuck exploring away from the target state. As such, an interesting direction for future work is to study exploration guarantees, for example, through regret bounds. Such analysis may lead to an automatic method for selecting the smallest δ that can guarantee exploration. A more simple (empirical) approach may quantify the rate at which the gating network's *epistemic uncertainty* is reducing and use it to relax δ if the agent has stopped exploring.

Myopic vs non-myopic active learning In dynamical systems, myopic learning corresponds to selecting the control input by only considering the information gain at the next state. In contrast, non-myopic learning selects the control input by considering the information gain over the next T states. The exploration strategy presented in this chapter selected the next T controls by considering the information gain over the next T states. In future work, it would be interesting to put

the exploration strategy into an MPC loop, such that it selects the next control, by considering the information gain over the next T states. This would enable a better comparison of myopic and non-myopic strategies.

Information criterion The exploration strategy presented in this chapter has shown that the GP-based gating network is useful for active learning when considering trajectories over a horizon. This is because the GPs have been able to model the joint distribution over the gating function values along a trajectory. This enabled the exploration strategy to find trajectories where each time step is aware of the information gain of other time steps. This is useful for finding non-myopic trajectories as it increases the information gain over the entire trajectory. However, this has only been verified by visual inspection and further analysis is left for future work. It is also worth noting that the exploration strategy is only possible because the gating network is based on GPs. As such, this method is not widely applicable in all MoGPE dynamics models.

An alternative approach is to use the entropy of the mode indicator variable α. Figure 6.9 compares how the gating function entropy and two alternative information-based objectives change during each ModeOpt iteration. Note that the figure shows myopic versions of the objectives that do not consider full trajectories. At the start of ModeOpt (top row), the shape of the entropy landscapes is quite similar. However, when the gating network starts learning the mode boundary at iteration $i = 4$ (second row) the shape of the entropy over the mode indicator variable (middle column) no longer matches the entropy of the gating function (left column). This is in line with what happens in GP classification and is to be expected from the gating network. As mentioned in Chapter 3, MoGPE gating networks tend to a uniform distribution 1) where they are not very confident (have high *epistemic uncertainty*) but also 2) where they are confident (have low *epistemic uncertainty*) but are at the boundary between modes. This can be seen in the last two rows of Figure 6.9. In this figure, high values (red) indicate regions where the objectives want to query next. The entropy of the mode indicator variable (middle column) is high at the mode

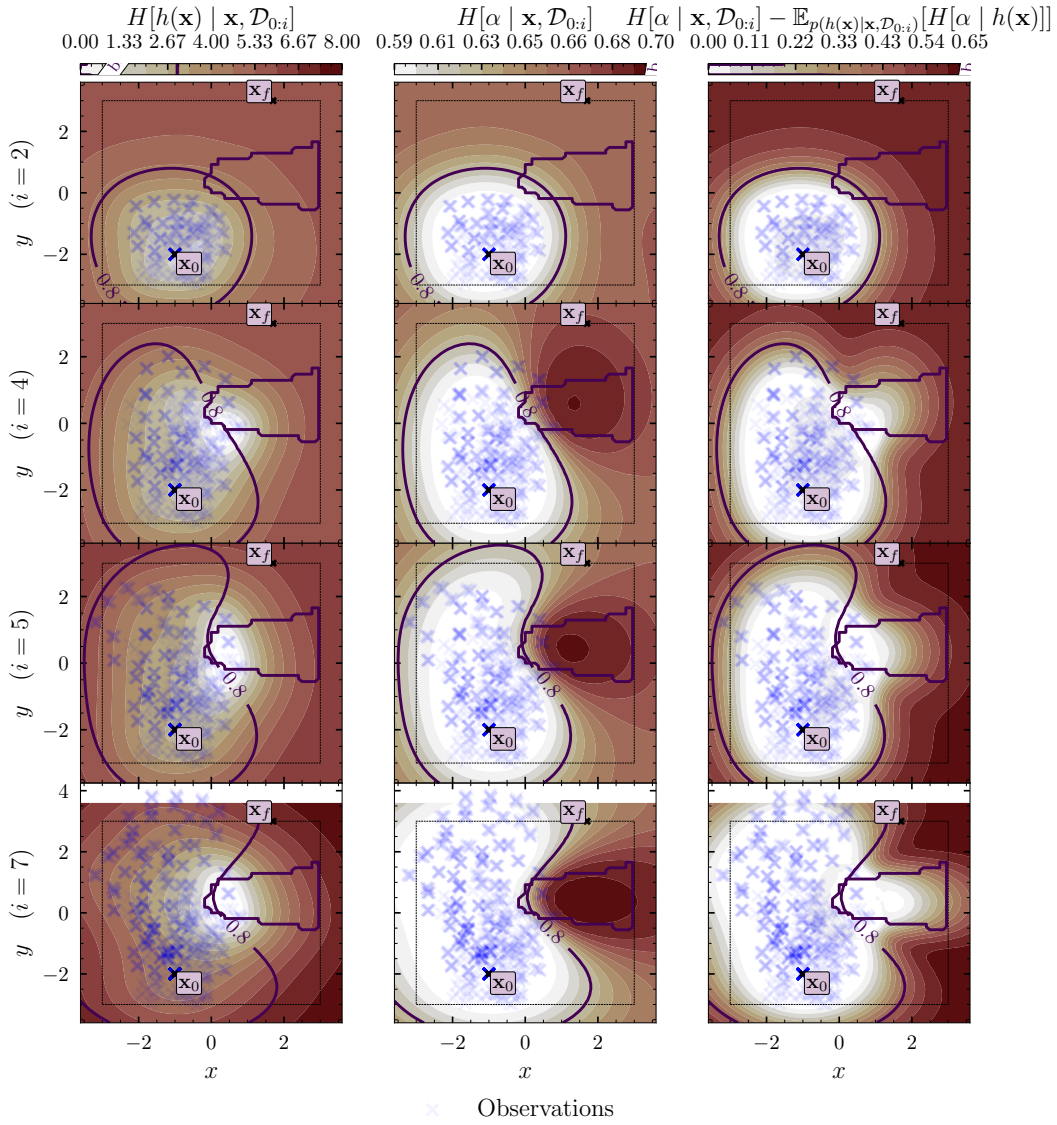


Figure 6.9: ***Comparison of information-based objectives at ModeOpt iterations*** $\mathbf{i = 2, 4, 5, 7}$ *Each column visualises how a different information-based objective changes as the dynamics model is updated during ModeOpt, i.e. after training on the data set* $\mathcal{D}_{0:i}$ *collected at previous iterations. Each row shows an iteration where the blue crosses indicate the data set* $\mathcal{D}_{0:i}$*. The left column shows the gating function entropy* $H[h(\mathbf{x}) \mid \mathbf{x}, \mathcal{D}_{0:i}]$ *(myopic version of the objective used in this work). The middle column shows the Bernoulli entropy of the mode indicator variable* $H[\alpha \mid \mathbf{x}, \mathcal{D}_{0:i}]$*. The right column shows the Bayesian Active Learning by Disagreement (BALD) objective which approximates* $H[\alpha \mid \mathbf{x}, \mathcal{D}_{0:i}] - \mathbb{E}_{p(h(\mathbf{x})|\mathbf{x},\mathcal{D}_{0,i})}[H[\alpha \mid h(\mathbf{x})]]$*. Note that this figure visualises the objectives at a single state and does not consider entire trajectories.*

boundary even though the model has observed data at this location. Therefore, this objective may keep exploring mode boundaries that it has already observed. In turn, this may lead to the strategy getting stuck in a local optimum at a mode boundary.

A popular approach to active learning for binary classification is Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011). BALD alleviates this issue by considering the following objective (in the myopic setting),

$$\arg\max_{\mathbf{x}_t} H(\alpha_t \mid \mathbf{x}_t, \mathcal{D}_{0:i}) - \mathbb{E}_{h_{k^*}(\mathbf{x}_t)\sim p(h_{k^*}(\mathbf{x}_t)|\mathcal{D}_{0:i})}[H(\alpha_t \mid \mathbf{x}_t, h_{k^*}(\mathbf{x}_t))]. \tag{6.13}$$

This objective is visualised in the right column of Figure 6.9. Intuitively, it seeks the state $\mathbf{x}_t$ for which the parameters under the posterior disagree about the outcome the most, hence the name. The objective is low in regions of the mode boundary which have been observed. This is a desirable behaviour which makes extending this objective to dynamical systems an interesting direction for future work.

It is worth noting that the joint distribution over the mode indicator variable in MoGPE models is factorised over time. As such, there is no way to condition the entropy at a given time step on all other time steps. However, it may be possible to use the joint distribution over the gating function values $p(h(\bar{\mathbf{x}}) \mid \bar{\mathbf{x}}, \mathcal{D}_{0:i})$ to achieve this behaviour.

More than 2 modes Although not tested here, ModeOpt is theoretically sound and should be applicable in environments with more than two dynamics modes. However, the exploration strategy uses the joint entropy of desired mode's gating function over a trajectory. Further experiments are required to see if this strategy works when the MoSVGPE dynamics model is instantiated with more than two experts. This is because when using more than two experts the MoSVGPE model uses the Softmax likelihood with a gating function for each expert. In contrast, the two expert case uses the Bernoulli likelihood with a single gating function.

Inducing points The method presented in this chapter initialised each of the sparse GPs in the MoSVGPE dynamics model with a fixed number of inducing points. Although this approach worked well in the experiments, it is unlikely that this will always be the case. For example, consider exploring much larger environments, i.e. environments with larger state domains. In these environments, the MoSVGPE's ability to accurately model an ever increasing data set with a fixed number of inducing points will decrease. This is due to the sparse approximation's ability to model the true nonparametric model deteriorating as the number of data points increases. See Burt et al., 2019 for details on rates of convergence for sparse GPs. As such, an interesting direction for future work is to study methods for dynamically adding new inducing points to each GP.

Fixing model parameters during training During its initial iterations, ModeOpt only explores the desired dynamics mode and does not observe any state transitions belonging to another mode. As a result, the lengthscale of the gating network GP increases. This often results in the gating network becoming overconfident, almost as if the gating network believes that only a single dynamics mode exists over the entire domain. When this happens, the $\delta-$mode remaining chance constraints expand the explorable region significantly further than the observed data. This is undesirable because it leads to trajectories leaving the desired dynamics mode significantly more. In practice, fixing the kernel's hyperparameters (e.g. lengthscale and signal variance) after training on the initial data set $\mathcal{D}_0$, appeared to alleviate this issue. However, it is left to future work to study this in more depth.

$\delta-$mode remaining Exploring a single dynamics mode in multimodal dynamical systems where the underlying dynamics modes and how the system switches between them, are *not fully known a priori*, is a hard problem. This is because the agent must observe regions outside of the desired dynamics mode in order to know that a particular region does not belong to the desired mode. The $\delta-$mode remaining exploration strategy proposed in Section 6.2.1 resulted in the agent leaving the desired dynamics mode multiple times in the experiments. In future work, it would

be interesting to study this in more detail. For example, how often must the agent leave the desired dynamics mode in order to accurately learn the mode boundary? And, how often does the agent fail when leaving the desired dynamics mode in practice?

Related work To the best of our knowledge, there is no previous work addressing the exploration of a single dynamics mode in multimodal dynamical systems. Schreiter et al., 2015 use a GP classifier to identify safe and unsafe regions when learning GP dynamics models in an active learning setting. However, they assume that they can directly observe whether a particular data point from the environment belongs to either the safe or unsafe regions. In contrast, we considered scenarios where the mode cannot be directly observed from the environment, but instead, is inferred by the probabilistic dynamics model.

6.5 Conclusion

This chapter has presented a novel strategy for exploring multimodal dynamical systems whilst remaining in the desired dynamics mode with high probability. Moreover, it has proposed how this exploration strategy can be combined with the dynamics model from Chapter 3, the $\delta-$mode remaining trajectory optimisation algorithms from Chapter 4 and the $\delta-$mode remaining chance constraints from Equation (4.41), to approximately solve the mode remaining navigation problem in Equation (2.4). That is, it can control multimodal dynamical systems – where the underlying dynamics modes and how the system switches between them, are *not fully known a priori* – from a start state $\mathbf{x}_0$, to a target state $\mathbf{x}_f$ whilst guaranteeing that the system remains in the desired dynamics mode with high probability.

The algorithm, named ModeOpt, was tested in a simulated version of the illustrative quadcopter example, verifying that the algorithm can work in practice. However, it has not been fully tested in environments with more than two modes, nor has it

been tested on a real-world system. Further testing and analysis of ModeOpt is left for future work.

7 Conclusion

The main objective of this thesis was to solve the mode remaining navigation problem in Equation (2.4). That is, to control a multimodal dynamical system – where neither the underlying dynamics modes, nor how the system switches between them, are *known a priori* – to a target state, whilst remaining in the desired dynamics mode. Based on well-established methods from Bayesian statistics and machine learning, this thesis proposed ModeOpt, a general framework for approximately solving the mode remaining navigation problem.

At the core of ModeOpt is a Bayesian approach to learning multimodal dynamical systems, named Mixtures of Sparse Variational Gaussian Process Experts (MoSVGPE), that accurately identifies the underlying dynamics modes, as well as how the system switches between them. Further to this, it learns informative *latent structure* that ModeOpt leverages to encode mode remaining behaviour into control strategies. The method's ability to learn factorised representations of multimodal data sets whilst retaining well-calibrated uncertainty estimates was validated on a real-world quadcopter data set, as well as on the motorcycle data set. Further to this, its applicability to learning dynamics models for model-based control was validated in two simulated environments.

As this thesis has focused on model-based techniques that leverage a learned dynamics model, it had to relax the requirement of remaining in the desired dynamics mode, to remaining in the desired dynamics mode with high probability. Initially, when not much of the environment has been observed, it is not possible to find trajectories to the target state that remain in the desired mode with high probability.

This is due to the learned dynamics model having high *epistemic uncertainty*. In this scenario, ModeOpt reduces the model's *epistemic uncertainty* by exploring the environment and updating the dynamics model with new data. ModeOpt sidesteps the exploration-exploitation trade-off which is common in MBRL algorithms by introducing a set of chance constraints. That is, ModeOpt does not need an objective function that changes its exploration-exploitation balance as it gathers more data. The mode chance constraints are a powerful tool that allow ModeOpt to deploy separate controllers during the explorative and exploration phases.

An explorative trajectory optimisation algorithm that leverages the GP-based gating network is proposed. Experiments confirm that the explorative controller is effective at maximising the information gain over the entire trajectory and targeting exploration towards the target state. However, further analysis of the exploration strategy is left for future work.

Three exploitative trajectory optimisation algorithms that find trajectories to the target state, whilst attempting to remain in the desired dynamics mode have been presented. Two of the methods show how the latent geometry of the GP-based gating network can be leveraged to encode mode remaining behaviour, whilst the third approach shows how this can be achieved by extending the CaI framework to multimodal dynamical systems. Their ability to remain in the desired dynamics mode was tested in two simulated environments. Both the Direct Optimal Control via Riemannian Energy (DRE) method in Section 4.2.3 and the Mode Remaining Control as Inference (MRCaI) method in Section 4.3 performed well in the experiments.

This thesis provided experimental evidence that ModeOpt can work in practice, however, this was only in simulation. Evaluating its performance on real-world problems is left for future work.

7.1 Future Work

There are many promising directions for future work. Some are extensions of the work in this thesis, whilst others are alternative approaches to solving the mode remaining navigation problem in Equation (2.4). Many of the extensions to the work in this thesis are discussed in the relevant chapters so are not re-discussed here.

Higher-dimensional problems Firstly, ModeOpt is mostly restricted to lower dimensional problems due to the difficulties of defining GP priors in high dimensions. In MBRL there has been interesting progress learning better statistical models and scaling them up to higher-dimensional problems, for example, using Bayesian neural networks. This is an interesting direction for future work as it may lead to more practical algorithms.

External sensing ModeOpt uses internal sensing to obtain information on the robot, such as where it is and how fast it is travelling. It then uses this information to infer the separation between the underlying dynamics modes from state transitions. However, in some applications, it may be possible to infer the separation between the underlying dynamics modes using external sensors. For example, in autonomous driving, the friction coefficients associated with different road surfaces may define a set of dynamics modes. In this setting, cameras could likely be used to detect changes in the road surface and thus the separation between the underlying dynamics modes. Although not applicable in all settings, this is a promising direction for future work, as the agent would never have to enter the undesired dynamics mode.

Real-time feedback control Although the trajectories found by the controllers are $\delta-$ mode remaining, often when they are executed in the environment they are not $\delta-$ mode remaining. This behaviour is expected with open-loop controllers. Although this did not pose an issue in the simulated experiments, it may pose an issue in real-world environments. Future work could explore methods for obtaining closed-loop controllers. A simple approach is to embed the trajectory optimisation

algorithm into an MPC loop to obtain a closed-loop controller. This would require the trajectory optimisation algorithms to be made faster so that they could be used for real-time MPC, for example, via locally linear dynamics approximations. Alternatively, an state-feedback controllers could be learned. For example, the trajectory optimisation algorithms could be used for guided policy search (Levine and Koltun, 2013), or, they could be used for policy optimisation by reformulating them as a sum of discounted rewards with the mode chance constraints implemented via Lagrange multipliers. It should be straightforward to verify that learned feedback controllers are $\delta -$ mode remaining under the dynamics using the mode chance constraints.

Model-free approaches Finally, this thesis has solely focused on model-based approaches for solving the mode remaining navigation problem in Equation (2.4). However, it may be possible to solve it with model-free methods. For example, Model-Free Reinforcement Learning (MFRL) methods may be able to learn reactive policies which automatically turn back when they encounter hard to control dynamics.

Bibliography

Ames, Aaron D., Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada (June 2019). "Control Barrier Functions: Theory and Applications". In: *2019 18th European Control Conference (ECC)*, pp. 3420–3431.

Anderson, T. W. (1946). "The Non-Central Wishart Distribution and Certain Problems of Multivariate Statistics". In: *The Annals of Mathematical Statistics* 17.4, pp. 409–431.

Ariafar, Setareh, Jaume Coll-Font, Dana Brooks, and Jennifer Dy (Jan. 2019). "ADMMBO: Bayesian Optimization with Unknown Constraints using ADMM". In: *Journal of machine learning research : JMLR* 20.

Auer, Peter (2002). "Using Confidence Bounds for Exploitation-Exploration Trade-offs". In: *Journal of Machine Learning Research* 3.Nov, pp. 397–422.

Bellman, Richard (1956). "Dynamic Programming". In: *Princeton University Press*.

Betts, John T. (Mar. 1998). "Survey of Numerical Methods for Trajectory Optimization". In: *Journal of Guidance, Control, and Dynamics* 21.2, pp. 193–207.

Boedecker, Joschka, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller (Dec. 2014). "Approximate real-time optimal control based on sparse Gaussian process models". In: *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–8.

Boney, Rinu, Norman Di Palo, Mathias Berglund, Alexander Ilin, Juho Kannala, Antti Rasmus, and Harri Valpola (2019). "Regularizing Trajectory Optimization with Denoising Autoencoders". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Buisson-Fenet, Mona, Friedrich Solowjow, and Sebastian Trimpe (2020). "Actively Learning Gaussian Process Dynamics". en. In: *2nd Annual Conference on Learning for Dynamics and Control*. Vol. 120. Proceedings of Machine Learning Research, pp. 1–11.

Burt, David, Carl Edward Rasmussen, and Mark Van Der Wilk (May 2019). "Rates of Convergence for Sparse Variational Gaussian Process Regression". en. In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, pp. 862–871.

Capone, Alexandre, Gerrit Noske, Jonas Umlauft, Thomas Beckers, Armin Lederer, and Sandra Hirche (2020). "Localized active learning of Gaussian process state space models". In: *2nd Annual Conference on Learning for Dynamics and Control*. Vol. 120:1-12. Proceedings of Machine Learning Research.

Carmo, Manfredo do (1992). *Riemannian Geometry*. en. Mathematics: Theory & Applications. Birkhäuser Basel.

Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". en. In: *Advances in Neural Information Processing Systems*. Vol. 31.

Cover M., Thomas and Thomas Joy A. (2006). *Elements of information theory*. John Wiley & Sons.

Deisenroth, Marc and Carl Rasmussen (Jan. 2011). "PILCO: A Model-Based and Data-Efficient Approach to Policy Search." In: *International Conference on Machine Learning*. Vol. 28, pp. 465–472.

Depeweg, S., J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft (2017). "Learning and policy search in stochastic dynamical systems with Bayesian neural networks". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Doerr, Andreas, CHristian Daniel, Duy Nguyen-Tuong, Alonso Marco, Stefan Schaal, Toussaint Marc, and Sebastian Trimpe (Oct. 2017). "Optimizing Long-term Predictions for Model-based Policy Search". en. In: *Conference on Robot Learning*. PMLR, pp. 227–238.

Dong, Jing, Mustafa Mukadam, F. Dellaert, and Byron Boots (2016). "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs". In: *Robotics: Science and Systems*.

Duffie, Darrell and Jun Pan (Feb. 1997). "An Overview of Value at Risk". en. In: *The Journal of Derivatives* 4.3, pp. 7–49.

Eduardo F., Camacho and Bordons Carlos (2007). *Model Predictive Control*. Springer.

Fahroo, F. and I. M. Ross (June 2000). "Direct trajectory optimization by a Chebyshev pseudospectral method". In: *Proceedings of the 2000 American Control Conference*. Vol. 6, pp. 3860–3864.

Ferber, R., R. Luce, and H. Raiffa (1958). *Games and Decisions: Introduction and Critical Survey*. Wiley New York.

Freeman, Randy and Petar V. Kokotovic (1996). *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*. en. Modern Birkhäuser Classics. Birkhäuser Basel.

Gal, Yarin, Rowan McAllister, and Carl Rasmussen (2016). "Improving PILCO with Bayesian Neural Network Dynamics Models". en. In.

Garg, Divya, Michael Patterson, William W. Hager, Anil V. Rao, David A. Benson, and Geoffrey T. Huntington (Nov. 2010). "A unified framework for the numerical solution of optimal control problems using pseudospectral methods". en. In: *Automatica* 46.11, pp. 1843–1851.

Gelbart, Michael A., Jasper Snoek, and Ryan P. Adams (2014). "Bayesian optimization with unknown constraints". English (US). In: *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014*. AUAI Press, pp. 250–259.

Ghahramani, Zoubin and Sam T. Roweis (1999). "Learning Nonlinear Dynamical Systems using an EM Algorithm". In: *Advances in Neural Information Processing Systems 11*. MIT Press, pp. 599–605.

Girard, Agathe (2004). "Approximate Methods for Propagation of Uncertainty with Gaussian Process Models". en. PhD thesis. University of Glasgow.

Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (May 2019). "Learning Latent Dynamics for Planning from Pixels". en. In: *International Conference on Machine Learning*. PMLR, pp. 2555–2565.

Hensman, James, Nicolo Fusi, and Neil D Lawrence (2013). "Gaussian Processes for Big Data". en. In: *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*. Vol. 29, pp. 282–290.

Hensman, James, Alexander Matthews, and Zoubin Ghahramani (Feb. 2015). "Scalable Variational Gaussian Process Classification". en. In: *Artificial Intelligence and Statistics*. PMLR, pp. 351–360.

Hewing, Lukas, Juraj Kabzan, and Melanie N. Zeilinger (Nov. 2020a). "Cautious Model Predictive Control Using Gaussian Process Regression". In: *IEEE Transactions on Control Systems Technology* 28.6, pp. 2736–2743.

Hewing, Lukas, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger (2020b). "Learning-Based Model Predictive Control: Toward Safe Learning in Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1, pp. 269–296.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John Paisley (2013). "Stochastic Variational Inference". In: *Journal of Machine Learning Research* 14.4, pp. 1303–1347.

Houlsby, Neil, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel (Dec. 2011). "Bayesian Active Learning for Classification and Preference Learning". In: *arXiv:1112.5745 [cs, stat]*.

Jacobs, Robert A., Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton (Mar. 1991). "Adaptive Mixtures of Local Experts". In: *Neural Computation* 3.1, pp. 79–87.

Janner, Michael, Justin Fu, Marvin Zhang, and Sergey Levine (2019). "When to Trust Your Model: Model-Based Policy Optimization". en. In: *Advances in Neural Information Processing Systems*. Vol. 32.

Kaiser, Markus, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek (Nov. 2020). "Bayesian decomposition of multi-modal dynamical systems for reinforcement learning". en. In: *Neurocomputing* 416, pp. 352–359.

Kamthe, Sanket and Marc Deisenroth (Mar. 2018). "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". en. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1701–1710.

Kappen, Hilbert J., Vicenç Gómez, and Manfred Opper (June 2013). "Optimal control as a graphical model inference problem". In: *Proceedings of the Twenty-Third International Conference on International Conference on Automated Planning and Scheduling*. ICAPS'13. Rome, Italy: AAAI Press, pp. 472–473.

Kelly, Matthew (Jan. 2017). "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation". en. In: *SIAM Review* 59.4, pp. 849–904.

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]*.

Kirk, Donald (2004). *Optimal control theory: an introduction*. Courier Corporation.

Koller, T., F. Berkenkamp, M. Turchetta, and A. Krause (Dec. 2018). "Learning-Based Model Predictive Control for Safe Exploration". In: *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066.

Krause, Andreas, Ajit Singh, and Carlos Guestrin (2008). "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies". In: *Journal of Machine Learning Research* 9.8, pp. 235–284.

Kurutach, Thanard, I. Clavera, Yan Duan, Aviv Tamar, and P. Abbeel (2018). "Model-Ensemble Trust-Region Policy Optimization". In: *ICLR*.

Kuss, Malte (2006). "Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning". en. PhD thesis. Technische Universität Darmstadt, Darmstadt, Germany.

Lambert, Nathan O., Daniel S. Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer S. J. Pister (Oct. 2019). "Low-Level Control of a Quadrotor With Deep Model-Based Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 4.4, pp. 4224–4230.

Levine, Sergey and Vladlen Koltun (May 2013). "Guided Policy Search". en. In: *International Conference on Machine Learning*. PMLR, pp. 1–9.

Ljung, Lennart (1999). *System Identification: Theory for the User*. 2nd. Prentice Hall Information and System Sciences Series. Pearson.

Loeliger, Hans-Andrea, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R. Kschischang (June 2007). "The Factor Graph Approach to Model-Based Signal Processing". In: *Proceedings of the IEEE* 95.6, pp. 1295–1322.

Lyapunov, A. M. (Mar. 1992). "The general problem of the stability of motion". In: *International Journal of Control* 55.3, pp. 531–534.

McKinnon, C. D. and A. P. Schoellig (May 2017). "Learning multimodal models for robot dynamics online with a mixture of Gaussian process experts". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 322–328.

Meeds, Edward and Simon Osindero (2006). "An alternative infinite mixture of gaussian process experts". In: *Advances in neural information processing systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press.

Moerland, Thomas, Joost Broekens, and Catholijn Jonker (2017a). "Efficient exploration with Double Uncertain Value Networks". In: *Neural Information Processing Systems*.

Moerland, Thomas M., Joost Broekens, and Catholijn M. Jonker (Aug. 2017b). "Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning". In: *arXiv:1705.00470*.

Mukadam, Mustafa, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots (Sept. 2018). "Continuous-time Gaussian process motion planning via probabilistic inference". en. In: *The International Journal of Robotics Research* 37.11, pp. 1319–1340.

Nagabandi, Anusha, Kurt Konolige, Sergey Levine, and Vikash Kumar (May 2020). "Deep Dynamics Models for Learning Dexterous Manipulation". en. In: *Proceedings of the Conference on Robot Learning*. PMLR, pp. 1101–1112.

Naish-guzman, Andrew and Sean Holden (2008). "The Generalized FITC Approximation". In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc.

Nakka, Yashwanth Kumar, Anqi Liu, Guanya Shi, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung (Apr. 2021). "Chance-Constrained Trajectory Optimization for Safe Exploration and Learning of Nonlinear Systems". In: *IEEE Robotics and Automation Letters* 6.2, pp. 389–396.

Nguyen-Tuong, Duy, Matthias Seeger, and Jan Peters (2009). "Model learning with local gaussian process regression". In: *Advanced Robotics* 23.15, pp. 2015–2034.

Parmas, Paavo, Carl Edward Rasmussen, Jan Peters, and Kenji Doya (July 2018). "PIPPS: Flexible Model-Based Policy Search Robust to the Curse of Chaos". en. In: *International Conference on Machine Learning*. PMLR, pp. 4065–4074.

Polymenakos, Kyriakos, Alessandro Abate, and Stephen Roberts (May 2019). "Safe Policy Search Using Gaussian Process Models". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1565–1573.

Pontryagin, Lev Semenovich (1987). *Mathematical theory of optimal processes*. CRC press.

Quinonero-Candela, Joaquin, A. Girard, J. Larsen, and C.E. Rasmussen (Apr. 2003). "Propagation of uncertainty in Bayesian kernel models - application to multiple-step ahead forecasting". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2, pp. II–701.

Quiñonero-Candela, Joaquin and Carl Edward Rasmussen (2005). "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *Journal of Machine Learning Research* 6.65, pp. 1939–1959.

Rasmussen, Carl and Zoubin Ghahramani (2001). "Infinite Mixtures of Gaussian Process Experts". en. In: *Advances in Neural Information Processing Systems*. Vol. 14, pp. 881–888.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. en. Adaptive computation and machine learning. Cambridge, Mass: MIT Press.

Rawlik, Konrad, Marc Toussaint, and Sethu Vijayakumar (2013). "On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference (Extended Abstract)". en. In: *Proceedings of the 23rd International Conference on Artificial Intelligence*, p. 5.

Rohr, Alexander von, Matthias Neumann-Brosig, and Sebastian Trimpe (May 2021). "Probabilistic robust linear quadratic regulators with Gaussian processes". en. In: *Learning for Dynamics and Control*. PMLR, pp. 324–335.

Rossi, Simone, Markus Heinonen, Edwin Bonilla, Zheyang Shen, and Maurizio Filippone (Mar. 2021). "Sparse Gaussian Processes Revisited: Bayesian Approaches to Inducing-Variable Approximations". en. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1837–1845.

Rybkin, Oleh, Chuning Zhu, Anusha Nagabandi, Kostas Daniilidis, Igor Mordatch, and Sergey Levine (June 2021). "Model-Based Reinforcement Learning via Latent-Space Collocation". In: *arXiv:2106.13229 [cs]*.

Sadigh, Dorsa and Ashish Kapoor (June 2016). "Safe Control Under Uncertainty with Probabilistic Signal Temporal Logic". en-US. In.

Scannell, Aidan, Carl Henrik Ek, and Arthur Richards (2021). "Trajectory Optimisation in Learned Multimodal Dynamical Systems Via Latent-ODE Collocation". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE.

Schneider, J. (1996). "Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning". In: *Advances in Neural Information Processing Systems*. Vol. 9, pp. 1047–1053.

Schreiter, Jens, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint (2015). "Safe Exploration for Active Learning with Gaussian Processes". en. In: *Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, pp. 133–149.

Schwarm, Alexander T. and Michael Nikolaou (1999). "Chance-constrained model predictive control". en. In: *AIChE Journal* 45.8, pp. 1743–1752.

Schön, Thomas B., Adrian Wills, and Brett Ninness (Jan. 2011). "System identification of nonlinear state-space models". en. In: *Automatica* 47.1, pp. 39–49.

Sekar, Ramanan, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak (Nov. 2020). "Planning to Explore via Self-Supervised World Models". en. In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR, pp. 8583–8592.

Shumway, R. and D. Stoffer (1982). "AN APPROACH TO TIME SERIES SMOOTHING AND FORECASTING USING THE EM ALGORITHM". In: *Journal of Time Series Analysis*.

Silverman, B W (1985). "Some aspects of the spline smoothing approach to non-parametric regression curve fitting". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 47.1, pp. 1–21.

Stengel, Robert F. (1986). *Stochastic optimal control: theory and application*. John Wiley & Sons, Inc.

Sutton, R.S. and A.G. Barto (2018). *Reinforcement learning, second edition: An introduction*. Adaptive computation and machine learning series. MIT Press.

Titsias, Michalis (Apr. 2009). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". en. In: *Artificial Intelligence and Statistics*. PMLR, pp. 567–574.

Tosi, Alessandra, Søren Hauberg, Alfredo Vellido, and Neil D Lawrence (2014). "Metrics for Probabilistic Geometries". en. In: *Proceedings of the 30th Conference*, pp. 800–808.

Toussaint, Marc (2009). "Robot Trajectory Optimization using Approximate Inference". In: *International Conference on Machine Learning*.

Toussaint, Marc and Amos Storkey (Jan. 2006). "Probabilistic inference for solving discrete and continuous state Markov Decision Processes". In: vol. 2006, pp. 945–952.

Tresp, Volker (2000). "Mixtures of Gaussian Processes". en. In: *Advances in Neural Information Processing Systems*. Vol. 13, pp. 654–660.

Vinogradska, Julia, Bastian Bischoff, Jan Achterhold, Torsten Koller, and Jan Peters (Jan. 2020). "Numerical Quadrature for Probabilistic Policy Search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.1, pp. 164–175.

Vinogradska, Julia, Bastian Bischoff, Duy Nguyen-Tuong, Anne Romer, Henner Schmidt, and Jan Peters (June 2016). "Stability of Controllers for Gaussian Process Forward Models". en. In: *International Conference on Machine Learning*. PMLR, pp. 545–554.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der

Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors (2020). "SciPy 1.0: Fundamental algorithms for scientific computing in python". In: *Nature Methods* 17, pp. 261–272.

Von Stryk, Oskar and Roland Bulirsch (Dec. 1992). "Direct and Indirect Methods for Trajectory Optimization". In: *Annals of Operations Research* 37, pp. 357–373.

Wang, Li, Evangelos A. Theodorou, and Magnus Egerstedt (May 2018). "Safe Learning of Quadrotor Dynamics Using Barrier Certificates". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2460–2465.

Watson, Joe, Hany Abdulsamad, Rolf Findeisen, and Jan Peters (May 2021). "Stochastic Control through Approximate Bayesian Input Inference". In: *arXiv:2105.07693 [cs, eess]*.

Yu, Hon Sum Alec, Dingling Yao, Christoph Zimmer, Marc Toussaint, and Duy Nguyen-Tuong (July 2021). "Active Learning in Gaussian Process State Space Model". In: *arXiv:2108.00819 [cs, stat]*.

Yuksel, Seniha Esen, Joseph N. Wilson, and Paul D. Gader (Aug. 2012). "Twenty Years of Mixture of Experts". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.8, pp. 1177–1193.

Ziebart, Brian D. (2010). "Modeling purposeful adaptive behavior with the principle of maximum causal entropy". PhD thesis.