The University of Sydney

Probabilistic Inference for Model Based Control

Lucas Barcelos de Oliveira

Supervisor: Prof. Fabio Ramos

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Faculty of Engineering
School of Computer Science

June, 2023

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

. . . .
Lucas Barcelos

Supervisor: Prof. Fabio Ramos PhD Candidate: Lucas Barcelos

Abstract

Robotic systems are crucial for increasing productivity, automation and to perform dangerous tasks. However, the complexity and unpredictability of physical systems requires robust and efficient control strategies that can manage uncertain parameters and adapt to changing environments. This thesis strives to push the state-of-the-art in robotic planning and control under uncertainty. We present an ensemble of learning-based methods to represent uncertain models, update our knowledge over their uncertainty in real-time and reason over multi-modal solutions.

In our first contribution, we introduce a framework which utilises modern techniques in Bayesian statistics to perform likelihood-free inference over the model parameters. This enables the use of complex black-box simulators to design a controller that is efficient and robust with respect to the uncertainty over simulation parameters. This approach employs the unscented transform and a variant of information theoretical model predictive control, providing a method that is efficient and robust over the uncertainty of simulation parameters. It demonstrates superior performance in evaluating trajectory rollouts in comparison to Monte Carlo sampling, reducing the online computation burden, as shown in a variety of control and robotics tasks.

Next we tackle the problem of multi-modal solutions by re-framing robotic planning and control as a Bayesian inference problem over the posterior distribution of actions and model parameters. To solve the inference problem, we propose an implicit variational inference algorithm, capable of estimating distributions over model parameters and control inputs in real-time by performing Stein Variational Gradient Descent. With this Bayesian formulation the method is able to approximate complex multi-modal posterior distributions, a requirement in dynamic and realistic robot navigation tasks.

Lastly, we address the problem of diversity in high-dimensional spaces. We discuss how the previous approach is prone to underestimating the uncertainty of the posterior distribution under certain conditions, generating solutions which are only locally optimal. Using advancements in the theory of rough paths, we devise an algorithm for parallel trajectory optimisation that promotes diversity in the solutions, hence avoiding mode collapse and ensuring better global properties. Our approach builds upon our variational inference method for trajectory estimation by employing diversity promoting kernels, leveraging the path signature representation of trajectories. Empirical tests on a variety of problems, from 2-D navigation to robotic manipulators operating in cluttered environments, underscore the method's efficiency, which surpasses that of the existing alternatives.

To the loves of my life.

Acknowledgments

Pursuing a Ph.D. has been a mesmerising ride, and I would not have made it through without the unwavering support of many dear friends and loved ones.

I'd like to begin by acknowledging the two people who, quite literally, made this journey possible. First, my most sincere thanks go to my wife and partner-in-crime, Fran. Your bravery to embark on this adventure with me is what made it a reality. Not many would board a plane with a 5-month-old baby to move to an unknown country on the other side of the world in pursuit of your partner's ambitious dream. Your unconditional love, support, and partnership carried me through every step of the way.

Secondly, my heartfelt appreciation goes to my supervisor, Prof. Fabio Ramos, who placed an immense amount of trust in me. Despite the geographic distance during most of my Ph.D. journey, his mentorship, creativity, and tenacity were instrumental in guiding my research and ensuring our work was the best it could possibly be. I am grateful for our many intriguing discussions, casual chats, the encouragement, and for maintaining such an open and friendly relationship.

I was fortunate to have two unofficial co-supervisors, Paulo Borges and Rafael Oliveira. Paulo, I appreciate your support, mentorship, and genuine concern for my well-being. You always managed to lift my spirits after our chats. And I have no words to express my gratitude to Rafa, who is essentially my de facto thesis co-supervisor. You are a force-of-nature, exceptionally bright, yet humble, and truly a kind soul. You both made me immensely proud of my roots.

I'm also profoundly grateful to all my lab colleagues and post-docs who shared this chapter of my life with me. I'd like to extend my gratitude to my good friend, programming whizz, and board game nemesis, Tin Lai, for all his assistance in our work together; the prolific William Zhi, for all the fascinating chats and cups of tea; Philippe Morere and Anthony Tompkins, who were early inspirations in this scientific journey; Lionel Ott and Gilad Francis, for showing me the ropes of the university's operations and organising numerous reading groups and seminars; Rafael Possas, who has always been exceptionally helpful and welcoming to me and my family; and finally, my new lab mates, Paco Tseng, Houston Warren, Wenzheng Zhang, and Kim Bente, thank you for reigniting the joy post-pandemic and keeping the flame alive. To all of you, thank you for indulging my insatiable appetite for board games; I hope you had as much fun as I did.

Lastly, I'd like to express my gratitude to my family. To Sinelma, my mother, and Gilson, my father, thank you for gifting me with such a happy childhood. Dad, I still remember you assisting me with my grade school assignments, your relentless pursuit of knowledge and pushing boundaries has always been a guiding light in my life. Mom, your lessons have shaped me in ways I can't put into words. I am also deeply grateful for the time you dedicated to supporting us in this endeavour. You've given so much from the heart, and from the bottom of my own, I thank you. To my uncle Jura, thank you for being my mentor, role model, and for loving me as your own son. And finally, thanks to my two beautiful boys, Nicolas and Benjamin, for filling our lives with joy and colour. To all of you, this work is as much yours as it is mine, and I'm sincerely grateful to have shared it with you.

Contents

List of Abbreviations

ABC	Approximate Bayesian Computation
ADP	Approximate Dynamic Programming
AGI	Artifical General Intelligence
AGV	Automated Guided Vehicle
AI	Artifical Intelligence
API	Application Programming Interface
BC	Behavioral Cloning
BGD	Batch Gradient Descent
CEM	Cross Entropy Method
CHOMP	Covariant Hamiltonian Optimisation for Motion Planning
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
DDP	Differential Dynamic Programming
DDPG	Deep Deterministic Policy Gradient
DISCO	Double Likelihood-free Inference Stochastic Control
DOF	Degrees of Freedom
DuSt-MPC	Dual Stein Variational Model Predictive Control
E-MDN	Ensemble of Mixture Density Networks
e.g.	*exempli gratia*, for example
ELBO	Evidence Lower Bound
GMM	Gaussian Mixture Model
GP	Gaussian Process
GPMP	Gaussian Process Motion Planing
GPU	Graphic Processing Unit

HJB	Hamilton-Jabobi-Bellman equation
i.e.	*id est*, that is
i.i.d.	independent and identically distributed
ICR	Inertial Centre of Rotation
iLQG	Iterative Linear Quadratic Gaussian
iLQR	Iterative Linear Quadratic Regulator
IT-MPC	Information Theoretical MPC
LFI	Likelihood Free Inference
LiDAR	Laser Imaging, Detection, and Ranging
LLM	Large Language Model
LQR	Linear Quadratic Regulator
MAP	*maximum a posteriori*
MCMC	Markov Chain Monte Carlo
MDN	Mixture Density Network
MDP	Markov Decision Process
ML	Machine Learning
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral
NLP	Non-linear Programming
NMPC	Non-linear Model Predictive Control
OC	Optimal Control
PDDP	Probabilistic Differential Dynamic Programming
PDE	Partial Differential Equation
PDF	Probability Density Function
PID	Proportional-Integral-Derivative
PS	Policy Search
RHC	Receding Horizon Control
RHS	right-hand side
RKHS	Reproducing Kernel Hilbert Space
RL	Reinforcement Learning
RRT	Rapidly-exploring Random Tree
s.t.	such that
SigSVGD	Kernel Signature Variational Gradient Descent

SMPC	Stochastic Model Predictive Control
SNMPC	Stochastic Non-linear Model Predictive Control
SOC	Stochastic Optimal Control
STOMP	Stochastic Trajectory Optimisation for Motion Planning
SVGD	Stein Variational Gradient Descent
SVMP	Stein Variational Motion Planning
SVMPC	Stein Variational Model Predictive Control
TRPO	Trust Region Policy Optimisation
UAV	Unmanned Aerial Vehicle
UT	Unscented Transform
VI	Variational Inference
w.r.t.	with respect to

Nomenclature

x	a scalar
i, j, k, t	indices
$\mathbf{x}$	a vector
$\mathbf{x}^\top$	the transpose of vector $\mathbf{x}$
$\mathbf{X}$	a matrix
$\mathbf{X}^{-1}$	inverse of matrix $\mathbf{X}$
$\mathbf{X}^\dagger$	generalised inverse of matrix $\mathbf{X}$
$\mathbf{I}$	the identity matrix
X	interchangeably, a path, trajectory, or sequence of scalar- or vector-valued variables
X	a random variable
$\mathbb{N}$	the set of natural numbers
$\mathbb{R}$	the field of real numbers
$\mathbb{R}^{m\times n}$	the set of m by n real matrices
$\mathcal{A} = \{\cdot\}$	a set
$f(\cdot)$	a function
$\mathrm{F}(\cdot)$	a functional whose domain are functions, as typical in Computer Science
$p(X \mid Y)$	the (conditional) Probability Density Function of random variable X given Y
$D_{\mathsf{KL}}(p \,\|\, q)$	Kullback-Leibler divergence between densities p and q
$\mathbb{E}[\cdot]$	expected value
$\mathbb{V}[\cdot]$	variance
$\mathrm{Cov}(X, Y)$	the covariance between random variables X and Y
$k(\mathbf{x}, \mathbf{y})$	kernel function evaluated at $\mathbf{x}$ and $\mathbf{y}$
$\boldsymbol{\mu}$	typically used to denote a mean vector
$\boldsymbol{\Sigma}$	typically used to denote a covariance matrix
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{U}(a, b)$	Uniform distribution in the closed interval $[a, b]$
$\mathcal{GP}$	a Gaussian Process

θ	parameters of a control policy
π_θ	a control policy parameterised by θ, i.e. $\pi(\cdot \mid \theta)$
ξ	parameters of a simulator, i.e. model of a dynamical system
$\propto$	proportional to
δ_a	the Dirac delta function evaluated at a, i.e. $\delta(x-a)$
$\mathbb{1}_{\mathcal{A}}(\cdot)$	indicator function of a set $\mathcal{A}$
$\mathcal{L}(\theta \mid X)$	log likelihood of parameters θ given the random variable X
$\mathcal{Q}$	configuration space, e.g. the space of parameters that define the configuration of a system
$\mathcal{W}$	work space in which a system operates, e.g. $\mathbb{R}^3$

List of Figures

List of Tables

Authorship Attribution

The contributions presented in this thesis have been published in the following conferences and journals, which form the core chapters of this thesis. The author's attribution includes, but is not limited to, the motivation, conceptualisation, formalisation, derivation, theorisation, experimentation, and communication of the following academic articles.

Chapter 3 appears as: Barcelos, L., Oliveira, R., Possas, R., Ott, L., & Ramos, F. (2020). DISCO: Double Likelihood-Free Inference Stochastic Control. *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 7. https://doi.org/978-1-7281-7395-5/20

Chapter 4 appears as: Barcelos, L., Lambert, A., Oliveira, R., Borges, P., Boots, B., & Ramos, F. (2021, July 12–16). Dual online stein variational inference for control and dynamics. In D. A. Shell, M. Toussaint & M. A. Hsieh (Eds.), *Robotics: Science and systems XVII (RSS)* (pp. 1–12). https://doi.org/10.15607/RSS.2021.XVII.068

Chapter 5 appears as: Barcelos, L., Lai, T., Oliveira, R., Borges, P., & Ramos, F. (2023). Path Signatures for Diversity in Probabilistic Trajectory Optimisation. [*manuscript submitted for publication*]. School of Computer Science, The University of Sydney

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

. . . .
Lucas Barcelos

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

. . . .

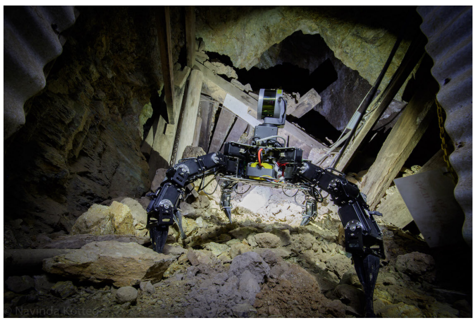
Prof. Fabio Ramos

The only constant in life is change.
On Nature
Heraclitus

CHAPTER 1

Introduction

Currently, we reside in the era of the scientific revolution, and despite a vocal anti-science minority, scientific knowledge has never played a more crucial role in human history than it does now. Our collective human challenges, such as global warming, space exploration, sanitation, and effective treatment for diseases, are just a few of the global issues we hope to solve through technological advancements. Recent progress in generative models like the Large Language Models (LLMs) chat-bots (Katz et al., 2023) and text-to-image models (Ramesh et al., 2021) have reignited the chase for Artifical General Intelligence (AGI), leading to renewed interest in robotics. Robotics is an interdisciplinary branch of technology that deals with the design, construction, operation, and use of robots. The aim is to create autonomous intelligent robots able to communicate with people and one another, learn from their experiences, and perform all kinds of dangerous or menial tasks (Lu et al., 2023).



(a) The hexapod robot Weaver which is capable of navigation in confined spaces, from Buchanan et al. (2019).

(b) An industrial fruit picking drone, from Tevel (2023).

(c) A companion robot, from Intuition Robotics (2023).

(d) A cleaning robot, from iRobot (2023).

Figure 1.1: Examples of robotics applications.

Despite their incredible success, most generative models are still a black-box science (Gillani, 2023). It is unclear how they reason and, importantly, how they deal with uncertain information. As the chapter epigraph emphasises, the only certainty we have is that nothing is certain. For all agents interacting with the world, uncertainty is unavoidable. Humans develop heuristics and surrogate models to adapt to changes in their environment. For example, individuals walk more carefully if the floor is slippery or drive more carefully during rainfall. However, adapting to uncertainty still poses a challenge for robots.

This thesis aims to explore how to represent and deal with uncertainty in robotic planning and control, including how to update behaviour as the uncertainty changes over time. Despite our focus on the direct application to robotics, the methods we will discuss are more general in nature and can be used to decision-making based on uncertain inductive models, such as financial and climate models.

1.1 Motivation

Robotics has become a valuable tool for increasing production and automating repetitive or hazardous tasks (Trevelyan, Hamel & Kang, 2016). Beyond automated manufacturing, robots have been used for construction (Burden, Caldwell & Guertler, 2022; Gharbia et al., 2020), domestic cleaning, and even companionship (Cooper et al., 2020). Some of these applications can be seen on Fig. 1.1. Furthermore, robots are equipped with a vast array of sensory inputs that allow them to perceive, navigate, and reason about the world (Wahrmann et al., 2016; Wittmann et al., 2015). However, it is important to note that each sensor has its own set of limitations. For example, cameras and laser ranging sensors are unable to detect objects obstructed by obstacles. Additionally, global positioning systems are susceptible to interferences caused by atmospheric conditions, multi-path reflections, and satellite visibility. Orientation sensors such as inertial measurement units are prone to generating estimates that drift over time, adding further uncertainty to models learned from sensor data.

The aforementioned signals are processed within a computational unit, which utilises the information gathered from the surrounding environment to perform reasoning and decision-making based on programmed instructions. Programming code is a fundamental aspect of robots, as it enables them to receive remote commands or operate autonomously by utilising artificial intelligence to interact with their environment. In this context, the term Artifical Intelligence (AI) refers to any manifestation of intelligence exhibited by machines, where the agent can sense its environment and execute actions to accomplish its objectives. Therefore, an artificially intelligent robot is an agent capable of displaying behaviors such as reasoning, planning, learning, perception, problem-solving, formal logic, and knowledge representation.

Typically, many of the tasks performed by robots rely on Machine Learning (ML) algorithms and techniques. ML is a type of artificial intelligence that enables machines to learn and improve their performance on a task without being explicitly programmed. It involves the use of algorithms and statistical models to analyse and draw insights from data, and then use those insights to make predictions or decisions about new data. In many cases breakthroughs are achieved with the use of extensive and labeled datasets, resource intensive neural networks or long iterative trial-and-error procedures. Although this paradigm has achieved many impressive results, they are of limited use when decisions need to be made in real-time under statistical models that evolve over time and where state-feedback creates an interplay between decisions and observations.

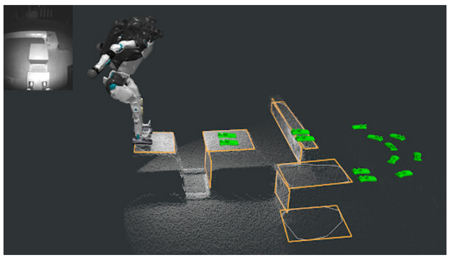
A synthetic and useful taxonomy for machine learning problems is the following:

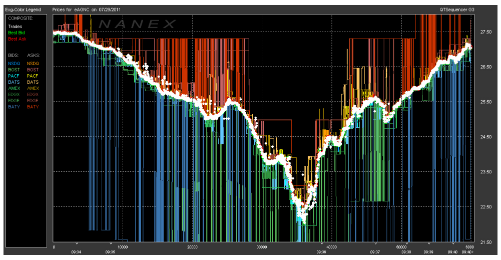
- *descriptive or unsupervised*: learn the distribution $p(z|x)$ that meaningfully summarises the latent variable z given observed data x;
- *predictive or supervised*: learn to predict an outcome y, given observed data x;
- *prescriptive or Reinforcement Learning (RL)*: prescribe an action a given a current state x to maximise a reward y.

As it can be seen, there is an intrinsic and progressive complexity to each of these tasks, where prescriptive analytics is the most challenging. In other words, descriptive analytics concerns making analysis in hindsight, predictive analytics focuses in determining what will happen, whereas prescriptive analytics is a foresight exercise of how an agent may interfere in the world to make a desirable outcome happen.

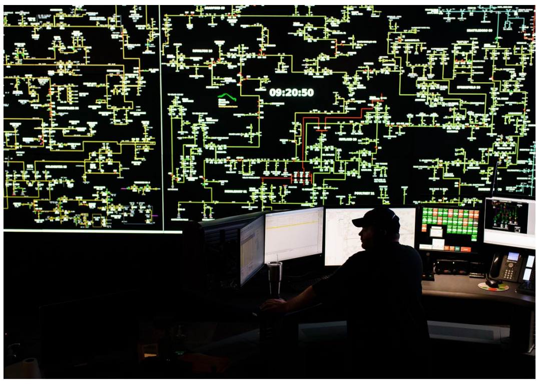
Robotic *control*, high-frequency algorithmic trading, and energy transmission are a few examples of prescriptive problems (Fig. 1.2). In computer science and engineering there are usually two distinct perspectives to address problems of this type, RL and Optimal Control (OC). In OC the goal is to derive complex actions from well-specified models with robustness guarantees to model uncertainties, whereas RL focuses on intricate model-free actions derived from data alone. In general lines, Optimal Control focuses primarily in continuous time and spaces and is based on physical models to derive robust control policies. Whereas RL starts from a discrete state-action space and tries to learn intricate policies from data alone. Those are, however, broad generalisations and in practice both fields have been extended beyond their original domains. In fact, for the specific cases of temporal difference algorithms with full Markov Decision Process (MDP), RL can be shown to be equivalent to an OC solution (Lewis, Vrabie & Vamvoudakis, 2012).

On the other hand, *planning* algorithms refer to a set of procedures designed to determine effective strategies for moving a system from an initial, suboptimal state to an improved state, commonly referred to as a goal state (LaValle, 2006). In the context


(a) Visualisation of the internal step control algorithm of Boston Dynamics's Atlas robot, from Kuindersma et al. (2016).



(b) Buy and sell signals from a high-frequency trading algorithm, from UC Berkeley D-Lab (2015).



(c) A power grid control room, from IEEE Spectrum (2023).

Figure 1.2: Examples of control applications.

of robotic systems, planners can direct the robot through its environment to accomplish specific tasks, such as mapping an environment. Empirical data obtained from observations allows us to validate and refine theoretical models that aim to describe the characteristics and behaviors of diverse physical environments. Simultaneously, these models assist the planning algorithm in navigating the robot through the environment, whether it involves avoiding hazardous regions, such as obstacles and challenging terrains, or identifying areas within the model that require additional data.

1.1.1 Model-free and model-based control

Strategies for solving these prescriptive problems can be coarsely subdivided in two main branches: *model-based* and *model-free* learning. In model-based Control, the first step is to find an approximate model of the system dynamics, which is then used in a feedback control loop to find an approximate solution. On the other hand, model-free control neglects the need of a dynamic model, searching for a policy directly from data. A policy is a map from the state of the environment to the actions to be taken and defines the learning agent's way of behaving at a given time. Formally, a policy π is a probability distribution over actions given states. In other words, given a state $\mathbf{x}$ and an action $\mathbf{u}$, $\pi(\mathbf{u} \mid \mathbf{x})$ is the probability that the agent will choose action $\mathbf{u}$ while in state $\mathbf{x}$.

The goal of many reinforcement learning algorithms is to find the optimal policy, which is the policy that will, from any initial state, yield the highest expected return, where the return is usually defined as the sum of rewards obtained after performing actions over some period of time. model-free methods can be subdivided into primarily two approaches: *Policy Search (PS)* and *Approximate Dynamic Programming (ADP)*. As expected, PS methods directly searches for policies by using data from previous episodes in order to improve the reward. On the other hand, ADP uses Bellman's principle of optimality to approximate a solution from previously observed data.

There is no consensus on the literature on which strategy is more successful or promising. Many researchers sustain that algorithms should be able learn innately how to derive a policy from experimentation alone, without access to the complex details required to simulate a dynamical system. The argument is that it will often be easier to directly find a policy for a task than to fit a general purpose model of the dynamics (Bertsekas, 2012). model-free approaches have been successful in a variety of tasks, however these approaches require a large number of iterations to optimise the policy and, sometimes, will also require expert demonstration to initiate learning. This implies on a large training dataset, which may constrain their application (Dann & Brunskill, 2015). However, when the system dynamics of high-dimensional spaces need to be learned, it is not clear whether an model approximation or value-function approximation yields better results (Atkeson & Santamaria, 1997).

1.1.2 Machine learning and robotic control

Nowadays machine learning and robotics are inextricably linked and one would be hard-pressed to find modern complex robots that do not rely on some form of machine learning to operate. Here we show a brief and non-exhaustive survey of how

RL and OC started to converge to the blurry outlines we have today. For a more extended survey on this topic, we refer the reader to the work of Recht (2018). Within model-based methods, results incorporating common OC methods like Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) have been used to solve RL problems. For instance, Abbeel, Coates and Ng (2010) used LQR in a RL setting to learn how to perform acrobatic maneuvers autonomously with an helicopter.

Another common model-based approach relies on MPC to either optimise a cost function or estimate a Q-function. Because MPC relies on Receding Horizon Control (RHC), the hypothesis is it will be more robust to disturbances and an effective way to achieve generalisation for RL tasks, which is particularly important in robotics. This idiosyncrasy is explored by Lowrey et al. (2019) to propose a framework where an MPC controller is used as a local trajectory optimiser in conjunction with a global value function approximator and an exploration planner. The authors propose a lemma showing that the cost increase due to model estimation errors is much lower for the MPC trajectory optimisation than that of a greedy policy improvement. Trajectory optimisation based on MPC has been used in prior work (Levine & Koltun, 2013; Mordatch & Todorov, 2014), but usually as a mean to improve PS.

At any rate, the approaches discussed above rely on a constrained optimisation, which implies that a convex approximation of the cost function and first or second order approximations of the dynamics are required (Camacho & Alba, 2013). Recently, Model Predictive Path Integral (MPPI) controllers have been successfully integrated to an RL framework achieving good results in several tasks, such as quadrotor trajectory control with obstacle avoidance and aggressive driving on a dirt track (G. Williams et al., 2017). MPPI is a trajectory sampling-based method, relying on the Free-Energy Principle (Friston, 2010) and Monte Carlo estimation to attain an asymptotically optimal trajectory for the length of the control horizon. As such, it is capable of handling complex cost criteria and is applicable to general nonlinear dynamics. Despite the use of importance sampling in (G. Williams et al., 2017) and the fact that MPPI requires no gradient updates, the task of sampling requires several forward passes on the neural network used to approximate the system dynamics and is in itself computationally expensive, posing as a bottleneck for its online application. Furthermore, despite its applicability to nonlinear dynamics, there are is no upper bound guarantees of robustness to model uncertainties and disturbances.

As for model-free methods, there has been a groundswell of activity in recent years. Levine and Koltun were part of the first to use MuJoCo as a test-bed and achieve robotic locomotion without special purpose techniques (Levine & Koltun, 2013). Since then, several results have been presented, as for example in (Lillicrap et al., 2016; Schulman et al., 2015; Silver et al., 2014).

However, despite the progress presented in these papers, model-free RL has been the target of substantial critic by some authors. At one hand, there are reservations on the Sample Efficiency of model-free RL for continuous problems. Intuitively, this may derive from the fact that at each update step, the model-free techniques we presented are improving on a single estimation of the policy or value function. In contrast, model-based methods iterate over n differential equations at each time step, where n is the dimension of the model used. This can be seen in practical results presented in (Dann & Brunskill, 2015; Henderson et al., 2018; Koch et al., 2019; Recht, 2018).

Another concern refers to the high variance of the methods to changes in intrinsic factors, such as the discount factor and random seeds (Henderson et al., 2018; Islam et al., 2017; Koch et al., 2019). This severely limits the reproducibility and or application of such approaches for most practical purposes, indicating there might be a bias towards bespoke solutions for achieving particular attractive results at benchmarks rather than a general purpose method. A perhaps surprising example of this is shown in (Koch et al., 2019), where a simple Proportional-Integral-Derivative (PID) controller yields better and more stable results than state-of-the-art algorithms, such as Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015) and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), for the straightforward task of altitude control of an Unmanned Aerial Vehicle (UAV), even after 10 million episodes of training.

1.2 Goals and research questions

Control and planning are crucial components of how robots behave and interact with the environment. Necessarily, this interaction is always imprecise and subject to randomness outcomes due to inaccuracies, noise and external disturbances. The study of how to best perform these tasks under uncertainty is discussed in the growing area of probabilistic robotics (Thrun, 2002). Out of a diversity of approaches on how to address stochasticity, in this thesis we are concerned on the problem of model uncertainty. In this setting, we are either given or learn a model that approximates the system dynamics and we incorporate uncertainties to the parameters of this model. This a natural way of quantifying uncertainty, associating it to physical attributes, and provides a useful framework with embedded inductive learning. In other words, we might have a better intuition for a prior distribution to capture the possible friction range on a mechanical joint than we might have for a latent representation of a Q-function.

Model uncertainty has been handled in many different ways, ranging from discounting future states estimation, establishing trust-regions for policies, the use of Gaussian Processes (GPs) to quantify uncertainty and combinations of these (Cai et al., 2021; Deisenroth, Fox & Rasmussen, 2015; Polydoros & Nalpantidis, 2017). One of the main challenges of learning policies for such systems is the limited availability of data, since it is usually costly and complex to assemble a comprehensive dataset. Researchers like Mandlekar et al. have tried addressing this problem through open access training platforms (Mandlekar et al., 2018), but those are still in incipient stages.

Furthermore, learning through simulated data is always limited to the capabilities of the simulator and the model inaccuracies, often referred to as *reality gap* (Jha & Lincoln, 2018). Additionally, given that robots interact with stochastic environments, we are interested in problems where the system dynamics are uncertain and/or evolve over time, a problem known as *covariate shift* (Ganegedara, Ott & Ramos, 2016). Therefore, we need not only to worry about learning a control policy, but also to incorporate model learning to account for such uncertainties.

Finally, despite their success in several applications, there are still many open problems that need to be addressed. Firstly, many of these approaches rely on gradients that are expensive to compute or sometimes unavailable. Additionally, in the typical case, the stochasticity is approximated with Gaussian noise in order to simplify the assumptions and result in a uni-modal solution. In reality, many problems are inherently ambiguous or naturally have multi-modal solutions in which case these approximations perform poorly (Lambert et al., 2020). Lastly, due to stochastic nature of the problem, it is difficult to incorporate constraints on actions and states.

In summary, the main goal of this thesis is to address the following high-level problem:

How should a robot behave when faced with imperfect information and ambiguous outcomes?

More specifically, we can formulate the following research questions:

- What is the best way to formulate uncertainty on model parameters and apply Bayesian inference for robotic control?
- How can we efficiently propagate model uncertainty in future state outcomes so as in to apply RHC?
- Can we improve the sample efficiency to allow learning uncertain models online? If so, what is the interplay between learning the model parameters and learning the control policy?
- In a probabilistic model formulation, is the control policy more amenable to sampling-based based methods over gradient-based optimisation?
- Can we efficiently capture the multi-modality of control and planning problems under uncertainty?

1.3 Contributions

The thesis consists of three contributing chapters, which are summarised as follows.

1.3.1 Efficiently Propagating Model Uncertainty with Posterior Moment-matching

Chapter 3 addresses the issue of how to formulate the control problem in a probabilistic framework and efficiently generate trajectory rollouts using a technique called Unscented Transform (UT). UT assumes that the posterior distribution is uni-modal and approximates the posterior up to a previously defined moment by evaluating a small set of sigma points. This idea is used to extend an MPPI controller to leverage prior knowledge of the model uncertainty and outperform the baseline case in a set of simulated and practical experiments. The chapter also discusses and perform ablation studies to evaluate the difference between using approximate gradients from Monte Carlo samples and numerical gradients.

1.3.2 Online learning the joint posterior distribution of control and model parameters

In Chapter 4 we reformulate the control problem of Chapter 3 in order to learn the complete posterior distribution over control actions. To accomplish this, the posterior is approximated with a series of particles which are in turn updated using Stein Variational Gradient Descent (SVGD). Additionally, the same approach is used to update distributions over the uncertain parameters of the model, allowing the agent to react to changes in the environment online. The chapter concludes by revisiting some of the experiments in Chapter 3 as well as including new experiments to evaluate the new method.

1.3.3 Tackling high-dimensional problems with path signatures

Finally, in Chapter 5, we discuss and tackle the issue of vanishing repulsive force when using SVGD in high-dimensional problems. The reducing repulsive force results in an underestimation of the posterior distribution, which leads to locally optimal solutions and mode collapse. We show how we can use a canonical feature representation of trajectories, the path signature, to reduce the problem dimensionality and generate solutions with better global properties. This is highlighted by experiments on a comprehensive motion planning benchmark for robotic manipulators, in which the proposed method outperforms other trajectory optimisation approaches, including that of Chapter 4.

1.4 Outline

This thesis is structured as follows. Following the introduction, three contributing chapters present novel methodologies discussed in Section 1.3. Chapter 2 provides the necessary theoretical background, covering Bayesian learning, Reproducing Kernel Hilbert Space, Stein Variational Gradient Descent, Optimal Control, Stochastic Non-linear Model Predictive Control, Model Predictive Path Integral, Likelihood Free Inference and Sim-to-real. Chapters 3 to 5 follow a consistent format, starting with an introduction, followed by a discussion of related work, methodology, and experiments. Finally, Chapter 6 concludes the thesis by reviewing the contributions and outlining directions for future research. Additional information for each of the main chapters is provided in Appendices A to C.

In the background of machine learning, lies a mosaic of ideas and breakthroughs, forming the bedrock upon which our present innovations are built.

Yoshua Bengio

CHAPTER 2

Background

2.1 Bayesian learning

Supervised learning involves learning a function that maps inputs to outputs based on a given dataset. Let our dataset $\mathcal{D}$ consist of N input-output pairs $(\mathbf{x}_i, y_i)$, where inputs are n-dimensional vectors from a set $\mathcal{X} \subset \mathbb{R}^n$, and outputs are real-valued scalars. In general, we will assume that the observed outputs are corrupted by i.i.d. noise η_i drawn from a given probability distribution. Our goal is to learn a function f that maps each input $\mathbf{x}_i$ to the corresponding output y_i and can predict the function value at unobserved inputs.

To find the best f, we assume that it belongs to a known hypothesis space $\mathcal{H}$ and try to find the function $f \in \mathcal{H}$ that best explains the data in $\mathcal{D}$. For instance, in the case of linear regression, we assume that

$$f(\mathbf{x}) = \boldsymbol{\omega}^{\mathsf{T}}\mathbf{x} \tag{2.1}$$

where $\boldsymbol{\omega}$ are the parameters of the model and the hypothesis space is the space of linear functions $\mathcal{H} = \{f : \mathcal{X} \to \mathbb{R} \mid f(\mathbf{x}) = \boldsymbol{\omega}^{\mathsf{T}}\mathbf{x}\}$. We can generalise this to the case of non-linear models by introducing a set of non-linear basis functions $\phi_i : \mathcal{X} \to \mathbb{R}$, $i \in \{1, \dots, q\}$, $q \in \mathbb{N}$, and assuming that

$$f(\mathbf{x}) = \boldsymbol{\omega}^{\mathsf{T}}\, \phi(\mathbf{x}), \tag{2.2}$$

where $\phi(\mathbf{x}) \coloneqq [\phi_1(\mathbf{x}), \dots, \phi_q(\mathbf{x})]$ is the vector of the basis functions' values at $\mathbf{x}$. Note that in this instance, the hypothesis space would no longer be linear, but rather a linear combination of the basis functions used.

The most common approach to finding the parameters $\boldsymbol{\omega}$ that best explain the data is to minimise a data-dependent loss function over the hypothesis space. However, this approach does not provide information about the model's confidence in its predictions given a limited amount of noise-corrupted data. In contrast, Bayesian learning provides a way to quantify model uncertainty by placing a belief distribution over the model and updating it based on the observed data. Predictions are then made by performing inference over the posterior distribution of the model conditioned on the data. In the case of parametric Bayesian learning, the posterior distribution over the model parameters $\boldsymbol{\omega}$ given the observed data $\mathcal{D}$ is given by Bayes' rule,

$$p(\boldsymbol{\omega} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\omega})\, p(\boldsymbol{\omega})}{p(\mathcal{D})}, \tag{2.3}$$

where $p(\boldsymbol{\omega})$ is the prior, $p(\mathcal{D} \mid \boldsymbol{\omega})$ is the likelihood of the parameters, and $p(\mathcal{D})$ is the evidence or marginal likelihood (MacKay, 2019). Since we now have a full posterior distribution $p(\boldsymbol{\omega} \mid \mathcal{D})$, we can assess how likely $\boldsymbol{\omega}$ is to be correct, encoding a confidence level in the model.

In Bayesian linear regression, a Gaussian prior is assumed over the parameters vector, $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\omega})$, with i.i.d. zero-mean Gaussian noise, $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$. The joint distribution between the data and parameters is then given by:

$$\begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{\Sigma}_{\omega} & \boldsymbol{\Sigma}_{\omega}\boldsymbol{\phi}^{\mathsf{T}} \\ \boldsymbol{\phi}\boldsymbol{\Sigma}_{\omega} & \boldsymbol{\phi}\boldsymbol{\Sigma}_{\omega}\boldsymbol{\phi}^{\mathsf{T}} + \sigma_\eta^2\mathbf{I} \end{bmatrix}\right), \tag{2.4}$$

where $\boldsymbol{\phi}$ is an n-by-q matrix with elements $\boldsymbol{\phi}_{ij} = \phi_j(\mathbf{x}_i)$ and $\mathbf{y}$ is a n-by-1 column vector with the given observations. We can obtain the posterior distribution in Eq. (2.3) by conditioning Eq. (2.4) on the observed data:

$$p(\boldsymbol{\omega} \mid \mathcal{D}) \sim \mathcal{N}(\boldsymbol{\Sigma}_{\omega}\, \boldsymbol{\phi}^{\top}\, \boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{y}, \boldsymbol{\Sigma}_{\omega} - \boldsymbol{\Sigma}_{\omega}\, \boldsymbol{\phi}^{\top}\, \boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\, \boldsymbol{\phi}\, \boldsymbol{\Sigma}_{\omega}), \tag{2.5}$$

where $\boldsymbol{\Sigma}_{\mathcal{D}} := \boldsymbol{\phi}\, \boldsymbol{\Sigma}_{\omega}\, \boldsymbol{\phi}^{\top}$. Given a query point $\mathbf{x}^{*}$, the value of f at that point can be predicted by the distribution

$$p(f(\mathbf{x}^{*}) \mid \mathcal{D}) = \mathcal{N}(m(\mathbf{x}^{*}), cov(\mathbf{x}^{*})) \tag{2.6}$$

$$m(\mathbf{x}^{*}) = \phi *^{\top}\boldsymbol{\Sigma}_{\omega}\, \boldsymbol{\phi}^{\top}\, \boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{y} \tag{2.7}$$

$$cov(\mathbf{x}^{*}) = \phi_{*}^{\top}\, \boldsymbol{\Sigma}_{\omega}\, \phi_{*} - \phi *^{\top}\boldsymbol{\Sigma}_{\omega}\, \boldsymbol{\phi}^{\top}\, \boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\, \boldsymbol{\phi}\, \boldsymbol{\Sigma}_{\omega}\, \phi_{*}, \tag{2.8}$$

where $\phi_{*} = \phi(\mathbf{x}^{*})$ and the mean of the distribution being the most likely prediction, and the variance quantifying the level of uncertainty.

As explained in (Rasmussen & Williams, 2006), $\phi(\mathbf{x})$ maps the n-dimensional vector $\mathbf{x}$ into a q-dimensional feature space, and these maps are used in the predictive distribution in the form of $(\mathbf{x}, \mathbf{x}') \mapsto \phi(\mathbf{x})^{\top}\boldsymbol{\Sigma}_{\omega}\boldsymbol{\omega}\, \phi(\mathbf{x}')$. This defines a positive-definite kernel or covariance function, $cov(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\top}\boldsymbol{\Sigma}_{\omega}\boldsymbol{\omega}\, \phi(\mathbf{x}')$, which allows feature maps to be abstracted to even infinite-dimensional feature spaces, leading to non-parametric methods.

2.2 Approximate Bayesian inference

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available. Bayesian inference is an important technique in statistics, and especially in mathematical statistics. In many practical applications, the exact Bayesian inference is computationally intractable. One of the main reason is because it requires computing the normalisation term

$$p(\mathbf{x}) = \int_{\theta} p(\mathbf{x} \mid \theta)\, p(\theta)\, \mathrm{d}\theta \tag{2.9}$$

which involves a high-dimensional integration over the parameter space. Approximate Bayesian inference methods are used to overcome this issue by computing approximations to the posterior distribution and the integrals involved in Bayesian inference.

There are two main approaches to approximate Bayesian inference, Variational Inference (VI) and Markov Chain Monte Carlo (MCMC).

2.2.1 Markov Chain Monte Carlo

MCMC is a stochastic approach to approximate Bayesian inference. The main idea is to construct a Markov chain that has the desired posterior distribution as its stationary distribution. Hence, by running the chain for a sufficiently long time, samples of the chain will coincide to samples of the target distribution (Haugh, 2021).

The transition probabilities of the Markov chain are defined such that the chain is guaranteed to converge to the desired distribution. Note that the desired distribution may only be know up to a factor and the stationary distribution of the Markov Chain is still guaranteed to converge to the (normalised) desired distribution. In the Metropolis-Hastings algorithm, a popular MCMC method, we start by defining a proposal distribution $h(x \mid X_n)$ that is easy to sample and will serve at suggesting transitions x. Then, at iteration n+1, the next state to be visited by the Markov Chain is defined by the following process. We start by drawing a suggested transition x from $h(x \mid X_n)$. The transition probability from state X_n to state x is then given by:

$$t(x \mid X_n) = a(x \mid X_n) \cdot h(x \mid X_n), \tag{2.10}$$

where the acceptance probability $t(\mathbf{x}' \mid \mathbf{x})$ is defined as:

$$a(x \mid X_n) = \min\left(1, \frac{p(x) \cdot h(X_n \mid x)}{p(X_n) \cdot h(x \mid X_n)}\right). \tag{2.11}$$

The acceptance probability ensures that states with higher posterior probability are more likely to be accepted. This way, the next state of the chain is given by

$$X_{n+1} = \begin{cases} x & \text{with probability } a \\ X_n & \text{with probability } 1 - a \end{cases} \tag{2.12}$$

MCMC methods are guaranteed to converge to the true posterior distribution given enough time, but they can be slow, especially in high-dimensional spaces. They also require careful tuning of the proposal distribution to ensure efficient sampling.

2.2.2 Variational Inference

Variational Inference is a deterministic approach to approximate Bayesian inference. The main idea is to turn the problem of computing the posterior distribution into an optimization problem. It does this by introducing a family of simpler, tractable distributions, denoted by $\mathcal{Q}$, and then finding the member of this family that is closest to the true posterior distribution (Blei, Kucukelbir & McAuliffe, 2017).

The closeness between the approximate distribution $q(\Theta)$ and the true posterior $p(\Theta \mid \mathbf{x})$ is measured using the Kullback-Leibler (KL) divergence:

$$q^*(\Theta) = \underset{q(\Theta)\in\mathcal{Q}}{\arg\min}\, D_{\mathrm{KL}}(q(\Theta) || \, p(\Theta \mid \mathbf{x})). \tag{2.13}$$

The KL divergence is a measure of the difference between two probability distributions. It is non-negative and equals zero if and only if the two distributions are the same. The optimization problem is typically solved using gradient-based methods. However, this is still a challenging task because it involves an intractable posterior. To overcome this, VI optimizes a lower bound to the log likelihood of the observed data, known as the Evidence Lower Bound (ELBO). The ELBO is derived by rearranging the KL divergence:

$$\log p(\mathbf{x}) = D_{\mathrm{KL}}(q(\Theta) || \, p(\theta \mid \mathbf{x})) + ELBO. \tag{2.14}$$

Since the KL divergence is always non-negative, the ELBO is a lower bound to the log likelihood of the observed data:

$$ELBO = \log p(\mathbf{x}) - D_{\mathrm{KL}}(q(\Theta) || \, p(\theta \mid \mathbf{x})). \tag{2.15}$$

The ELBO can also be written in terms of expectations with respect to the variational distribution $q(\Theta)$:

$$ELBO = \mathbb{E}_{q(\Theta)}[\log p(\mathbf{x}, \Theta)] - \mathbb{E}_{q(\Theta)}[\log q(\Theta)], \tag{2.16}$$

where first term on the right-hand side is the expected log joint probability, and the second term is the entropy of the variational distribution. The ELBO is a function of the variational parameters, and VI optimizes the ELBO with respect to these parameters. By maximizing the ELBO, we are pushing up on the lower bound to the log likelihood of the observed data, and indirectly minimizing the KL divergence between the variational distribution and the true posterior. This results in an approximation to the posterior that is as close as possible to the true posterior, given the family of distributions that we have chosen for $q(\theta)$. Therefore, contrary to sampling approaches, in VI a model is assumed (the parameterised distribution family), implying a bias but also a lower variance. As such, the choice of the family $\mathcal{Q}$ is crucial in variational inference. A common choice is the mean-field variational family, where each latent variable is assumed to be independent and is associated with its own variational parameter.

Both Variational Inference and MCMC have their strengths and weaknesses. VI is typically faster and can work better for large datasets, but the quality of the approximation can be hard to assess, and the choice of the family $\mathcal{Q}$ can significantly affect the results. MCMC, on the other hand, is asymptotically exact, but it can be slow and requires careful tuning of the proposal distribution. The choice between VI and MCMC often depends on the specific problem and the trade-off between computational efficiency and approximation quality.

2.3 Likelihood-free parameter estimation

Nowadays simulation is extensively used in a variety of scientific domains, such as protein folding, population genetics, particle physics, etc. This has been fueled by the advancements in computing power, Graphic Processing Units (GPUs), parallelism, and the expressiveness of modern computing languages. The wide availability of these tool have made the development of highly complex high-fidelity simulations more accessible than ever before. Unfortunately, one significant drawback of such complex simulators is their unsuitability for statistical inference. The main issue is that the probability density, or even the likelihood, for a given observation is typically intractable. This is a crucial requirement for both frequentist and Bayesian inference methods. Hence why these problems are commonly referred to as implicit models, in contrast to the prescribed models in which the likelihood of an observation can be explicitly calculated.

In Likelihood Free Inference (LFI), the main goal is to estimate posterior distributions from data on systems where the likelihood is intractable. In a typical scenario, a Likelihood-free inference method takes a prior $p(\boldsymbol{\xi})$ over simulation parameters $\boldsymbol{\xi}$, a black box generative model or simulator $\mathbf{x} = g(\xi)$ that generates simulated observations $\mathbf{x}$ from these parameters, and observations from the physical world $\mathbf{x}^r$ to compute the posterior $p(\xi \mid \mathbf{x}, \mathbf{x}^r)$. The main difficulty in computing this posterior relates to the evaluation of the likelihood function $p(\mathbf{x} \mid \xi)$ which is defined implicitly from the simulator (Diggle & Gratton, 1984). Here we assume that the simulator is a set of dynamical differential equations associated with a numerical or analytical solver which are typically intractable and expensive to evaluate. Furthermore, we do not assume these equations are known and treat the simulator as a black box. This allows LFI methods to be utilised with many robotics simulators (even closed source ones) but requires a method where the likelihood cannot be evaluated directly but instead only sampled from, by performing forward simulations. The most popular family of algorithms to address it are known as Approximate Bayesian Computation (ABC) (Beaumont, Zhang & Balding, 2002), and it's variants (Bonassi & West, 2015; Marjoram et al., 2003; Pritchard et al., 1999).

Recent advances in LFI allowed the use of probabilistic inference to learn distributions over simulation parameters (Ramos, Possas & Fox, 2019). The main idea is that of approximating an intractable posterior $p(\xi \mid \mathbf{x}, \mathbf{x}^r)$ using data generated from a generative forward model (or simulator) where trajectories are collected for different simulation configurations. Therefore, one can directly learn a conditional density $q_\phi(\xi \mid \mathbf{x})$ where parameters ϕ are learned through an optimisation procedure. The learned model usually takes the form of a mixture of Gaussians where inputs are summary statistics obtained from trajectories and outputs are the parameters of the mixture.

The goal is to maximise the likelihood $\prod_n q_\phi(\xi_n \mid \mathbf{x}_n)$. It has been shown in previous work (Ramos, Possas & Fox, 2019) that $q_\phi(\xi \mid \mathbf{x})$ will be proportional to

$$\frac{\tilde{p}(\xi)}{p(\xi)}\, p(\xi \mid \mathbf{x}) \tag{2.17}$$

if the log-likelihood is optimised with the following loss:

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_n \log q_\phi(\xi_n \mid \mathbf{x}_n). \tag{2.18}$$

Consequently, a posterior estimate can be obtained by

$$\hat{p}(\xi \mid \mathbf{x} = \mathbf{x}^r) \propto \frac{p(\xi)}{\tilde{p}(\xi)}\, q_\phi(\mathbf{x} = \mathbf{x}^r). \tag{2.19}$$

The conditional density $q_\phi(\boldsymbol{\xi} \mid \mathbf{x})$ is a mixture of K Gaussians,

$$q_\phi(\boldsymbol{\xi} \mid \mathbf{x}) = \sum_{k=1}^{K} \alpha_k(\mathbf{x}) \mathcal{N}(\boldsymbol{\xi} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.20}$$

where $\{\alpha_k(\mathbf{x})\}_{k=1}^K$ are mixing functions, $\{\boldsymbol{\mu}_k\}_{k=1}^K$ are mean functions and $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ are covariance functions.

2.4 Unscented Transform

The UT is a map used to estimate the result of applying a given nonlinear transformation to a probability distribution that is characterised only in terms of a finite set of statistics As such, it is able to reconstruct an approximate Y' of the random variable $Y = g(X)$ resulting when an original random variable X is applied to a non-linear function f. The premise behind this approach is that it should be easier to approximate a probability distribution than a arbitrary non-linear transformation. The idea is to select a set of *sigma points* able to capture the most important statistical properties of the prior random variable X. The necessary statistical information captured by the UT is the first and second order moments of $p(X)$. The number of sigma-points needed to do this $L = 2n + 1$, where n is the dimension of X. Van der Merwe (2004) has shown that matching the moments of X up to the nth order implies matching the moments of Y to the same order. By using a larger number of sigma-points, skew and kurtosis can also be captured (Julier, 2002). Note however, that this moment-matching approach provides an uni-modal approximation to the potentially multi-modal true distribution.

The expressions to compute the sigma-points and weights for the mean, ϖ_0^m, and covariance, ϖ_0^c, of the distribution $p(\boldsymbol{\xi})$ are presented below (van der Merwe, 2004):

$$\begin{aligned}
\varpi_0^m &= \frac{\nu}{n+\nu} & \chi_0 &= \boldsymbol{\mu}_\xi \\
\varpi_i^m = \varpi_i^c &= \frac{1}{2(n+\nu)} & \chi_i &= \boldsymbol{\mu}_\xi + \left(\sqrt{(n+\nu)\boldsymbol{\Sigma}_\xi}\right), \text{ for } 1 \le i \le n \\
\varpi_0^c &= \varpi_0^m + \left(1 - \nu^2 + o\right) & \chi_i &= \boldsymbol{\mu}_\xi - \left(\sqrt{(n+\nu)\boldsymbol{\Sigma}_\xi}\right), \text{ for } n+1 \le i \le 2n,
\end{aligned} \tag{2.21}$$

where $\nu = \alpha^2(n + \kappa) - n$ is the primary scaling factor, κ a secondary scaling (usually zero), α determines the spread of the sigma points around $\boldsymbol{\mu}_\xi$, and o is a scalar to provide an extra degree freedom. The reader is encouraged to refer to van der Merwe (2004) for details on hyper-parameter selection.

2.5 Reproducing kernel Hilbert spaces

The projection of inputs into a possibly infinite-dimensional feature space, as seen in Section 2.1, is intimately related with the concept of a Reproducing Kernel Hilbert Space (RKHS). Essentially, a positive-definite covariance function defines a reproducing kernel and an associated Hilbert space. In this section we will revise these concepts, which will play an important role in deriving models and theoretical results in subsequent chapters. We begin by reviewing linear spaces, which are a superclass of Hilbert spaces, and follow by defining reproducing kernels and RKHS's in Section 2.5.2. For a more comprehensive understanding, readers can refer to the extensive literature on the topic of RKHS's, including (Schölkopf & Smola, 2002) and (Steinwart & Christmann, 2008).

2.5.1 Linear spaces

In summary, a Hilbert space is a complete inner product space with respect to the norm induced by its inner product Although Hilbert spaces can also be defined for vector-valued functions, this section will focus on real, scalar-valued Hilbert spaces. In addition, this section briefly covers some related concepts of mathematical spaces that are necessary for a more comprehensive definition of a Hilbert space and a RKHS, and further information on these topics can be found in books on functional analysis, such as (Lax, 2014).

For simplicity, we will assume that the field $\mathbb{F}$ in which the linear spaces are defined is the $\mathbb{R}$, that any $f, g, h \in \mathcal{V}$, and any $\alpha, \beta \in \mathbb{R}$.

Definition 1 (Vector spaces) A vector space, also known as a linear space, is a non-empty set $\mathcal{V}$ that is closed under the operations of addition and scalar multiplication. Closure of an operation on a set means that performing the operation on elements of the set always results in an element of the same set. For instance, $\mathbb{R}^d$, where $d \in \mathbb{N}$, represents the space of d-dimensional real-valued vectors, and is a vector space. Similarly, the space $\mathbb{R}^{\mathcal{X}}$, consisting of all real-valued functions $f : \mathcal{X} \to \mathbb{R}$ on any non-empty set $\mathcal{X}$, is also a vector space when addition and scalar multiplication operations are defined as follows:

$$(f+g)(x) = f(x) + g(x)\,,\ \forall x \in \mathcal{X}\,, \tag{2.22}$$

$$(\alpha f)(x) = \alpha f(x)\,,\ \forall x \in \mathcal{X}\,, \tag{2.23}$$

and satisfy the axioms:

VS1. $f + (g + h) = (f + g) + h$

VS2. $\exists 0 \in \mathcal{V}$ s.t. $0 + f = f$

VS3. $\forall f \in \mathcal{V} \exists f' \in \mathcal{V}$ s.t. $f + f' = 0.$

VS4. $f + g = g + f$

VS5. $1f = f, \quad \alpha(\beta f) = (\alpha\beta)f, \quad (\alpha+\beta)f = \alpha f + \beta f, \quad \alpha(f+g) = \alpha f + \alpha g$

Definition 2 (Inner product spaces) An inner product space $(\mathcal{V}, \langle\cdot,\cdot\rangle_{\mathcal{V}})$ is a vector space $\mathcal{V}$ equipped with an inner product, denoted by the operator $\langle\cdot,\cdot\rangle_{\mathcal{V}} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$, that satisfies the following axioms:

IP1. $\langle f, g\rangle_{\mathcal{V}} = \langle g, f\rangle_{\mathcal{V}}$;

IP2. $\langle f, f\rangle_{\mathcal{V}} \geq 0, \quad \langle f, f\rangle_{\mathcal{V}} = 0 \iff f = 0$;

IP3. $\langle \alpha f + \beta g, h\rangle_{\mathcal{V}} = \langle \alpha f, h\rangle_{\mathcal{V}} + \langle \beta g, h\rangle_{\mathcal{V}} = \alpha\langle f, h\rangle_{\mathcal{V}} + \beta\langle g, h\rangle_{\mathcal{V}}$.

The definition of the inner product also implies in the following form of the *Cauchy-Schwartz* inequality:

$$|\langle f, g\rangle_{\mathcal{V}}| \leq \sqrt{\langle f, f\rangle_{\mathcal{V}}\langle g, g\rangle_{\mathcal{V}}}. \tag{2.24}$$

A simple example of an inner product space is $\mathbb{R}^d$, $d \in \mathbb{N}$, equipped with the dot product operation:

$$\langle \mathbf{x}, \mathbf{y}\rangle_{\mathbb{R}^d} := \mathbf{x} \cdot \mathbf{y} := \sum_{i=1}^{d} x_i y_i,\ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{2.25}$$

Definition 3 (Normed vector spaces) A normed vector space $(\mathcal{V}, \|\cdot\|_{\mathcal{V}})$ is any vector space $\mathcal{V}$ that contains a norm $\|\cdot\|_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}_0^+$, which assigns a non-negative length or size to each vector in the space. A norm can be any function satisfying:

N1. $\|\alpha f\|_{\mathcal{V}} = |\alpha| \|f\|_{\mathcal{V}}$;

N2. $\|f\|_{\mathcal{V}} \geq 0, \quad \|f\|_{\mathcal{V}} = 0 \iff f = 0$;

N3. $\|f + g\|_{\mathcal{V}} \leq \|f\|_{\mathcal{V}} + \|g\|_{\mathcal{V}}$.

Examples of norms applied in this thesis are p-norms, such as:

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{d} x_i^p \right)^{\frac{1}{p}}, \ \forall \mathbf{x} \in \mathbb{R}^d, \tag{2.26}$$

which are valid norms for $p \geq 1$, with $\|\mathbf{x}\|_\infty := \max x_i$, for $i \in \{1, \dots, d\}$. Similarly, for real-valued functions over some domain $\mathcal{X} \subset \mathbb{R}^d$, i.e. $\mathcal{V} = \mathbb{R}^{\mathcal{X}}$, one can define:

$$\|f\|_p := \left(\int_{\mathcal{X}} |f(\mathbf{x})|^p \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}. \tag{2.27}$$

However, it should be noted that the $\|\cdot\|_p$ norm may not be positive-definite on the entire $\mathbb{R}^{\mathcal{X}}$. For instance, a function $f \in \mathbb{R}^{\mathcal{X}}$ that is zero almost everywhere, except for a countable set of points in $\mathcal{X}$, would still yield $\|f\|_p = 0$, even though $f \neq 0$. Moreover, by definition, any norm must be finite, since ∞ is not an element of $\mathbb{R}_0^+$. Nevertheless, if we restrict ourselves to $\mathcal{V} = \mathcal{C}(\mathcal{X})$, which is the set of continuous real-valued functions on a compact $\mathcal{X} \subset \mathbb{R}^d$, then $\|\cdot\|_p$ becomes a valid norm on elements of that set.

Note that from an inner product we can define a induced norm given by: $\|\cdot\|_{\mathcal{V}} := \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{V}}}$. Using the inner product defined in Eq. (2.25), is equivalent to the 2-norm as defined in Eq. (2.26), where $p = 2$.

Definition 4 (Metric spaces) A metric space $(\mathcal{V}, d_{\mathcal{V}})$ is simply a set $\mathcal{V}$ equipped with a metric $d_{\mathcal{V}}$. A metric on a set $\mathcal{V}$ is any function $d_{\mathcal{V}} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_0^+$, where $\mathbb{R}_0^+$ denotes the set of non-negative real numbers, which satisfies the following axioms:

M1. $d_{\mathcal{V}}(f, g) = d_{\mathcal{V}}(g, f)$;

M2. $d_{\mathcal{V}}(f,g) \geq 0, \quad d_{\mathcal{V}}(f,g) = 0 \iff f = g;$

M3. $d_{\mathcal{V}}(f,g) \leq d_{\mathcal{V}}(f,h) + d_{\mathcal{V}}(h,g).$

Note that $\mathcal{V}$ does not need to be a vector space to have a defined metric. Also, both normed vector spaces and inner product spaces have, by definition, metrics associated with them. To see this, consider that for a given norm $\|\cdot\|_{\mathcal{V}}$, the induced metric is:

$$d_{\mathcal{V}}(f,g) := \|f - g\|_{\mathcal{V}}, \ \forall f,g \in \mathcal{V}. \tag{2.28}$$

Finally, the metric space defined by $\mathbb{R}^d$, $d \in \mathbb{N}$, and $\|\cdot\|_2$ is referred to as an *Euclidean vector space*. As the equation above still defines a valid metric if one scales the vectors by a positive-definite matrix $\mathbf{M}$, a non-Euclidean metric on $\mathbb{R}^d$ is the *Mahalanobis distance* and its associated norm, given by:

$$\|\mathbf{x} - \mathbf{y}\|_{\mathbf{M}} := \sqrt{(\mathbf{x} - \mathbf{y})^{\mathsf{T}}\mathbf{M}(\mathbf{x} - \mathbf{y})}. \tag{2.29}$$

Definition 5 (Complete metric spaces) A metric space is complete if the limit of every Cauchy sequence in it is an element of the same space (Munkres, 2014). A sequence $\{f_i\}_{i\in\mathbb{N}}$ in a normed vector space $(\mathcal{V}, \|\cdot\|_{\mathcal{V}})$ is Cauchy if, and only if, for every $\epsilon > 0$, there is a $n_\epsilon \in \mathbb{N}$, such that:

$$\forall i,j > n_\epsilon, \ \|f_i - f_j\|_{\mathcal{V}} < \epsilon. \tag{2.30}$$

Without completeness it is not possible to ensure that the limit of any given convergent sequence is an element of the same metric space. That is why important concepts in calculus and functional analysis are only valid for complete metric spaces. As particular types of complete metric spaces, we have Banach spaces and Hilbert spaces. It is also known that every metric space has a completion (Kreyszig, 1989).

Definition 6 (Banach spaces) A normed vector space that is complete with respect to the metric induced by its norm is called a Banach space.

Definition 7 (Hilbert spaces) A Hilbert space is an inner product space that is complete with respect to the norm induced by its inner product.

By its definition, we can see that any Hilbert space is also a Banach space with norm defined by the inner product. As a trivial example of a Hilbert space, we have Euclidean vector spaces $\mathbb{R}^d$. Hilbert spaces are a powerful tool in machine learning thanks to the concept of reproducing kernels, which is explained in the next section.

2.5.2 Reproducing kernels and their Hilbert spaces

Definition 8 (Reproducing kernel Hilbert space) Let $\mathcal{X}$ be a non-empty set and let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Let $cov : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function such that for any $\mathbf{x} \in \mathcal{X}$, the function $cov(\mathbf{x}, \cdot) : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space with reproducing kernel cov if for any finite sequence of points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q \in \mathcal{X}$ and coefficients $c_1, c_2, \dots, c_q \in \mathbb{R}$ the following axioms hold:

RK1. (Positive-definiteness) The kernel cov is a positive-definite function, or equivalently:

$$\sum_{i=1}^{q} \sum_{j=1}^{q} c_i c_j \, cov(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

RK2. (Linearity) The kernel cov is linear in its second argument, that is:

$$k\left(\mathbf{x}, \sum_{i=1}^{q} c_i \mathbf{y}_i\right) = \sum_{i=1}^{q} c_i \, k(\mathbf{x}, \mathbf{y}_i)$$

RK3. (Reproducing property) For any $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$, we have:

$$f(\mathbf{x}) = \langle f, cov(\mathbf{x}, \cdot) \rangle_{\mathcal{H}}.$$

These axioms imply that the kernel cov uniquely determines the Hilbert space $\mathcal{H}$, and that $\mathcal{H}$ is a complete inner product space with respect to the norm induced by the inner product:

$$\|f\|_2 = \langle f, f \rangle = \sum_{i=1}^{q} \sum_{j=1}^{q} c_i c_j \, cov(\mathbf{x}_i, \mathbf{x}_j) \tag{2.31}$$

The relationship between positive-definite kernels and RKHS's is bidirectional. Firstly, any Hilbert space of functions whose evaluation functional is continuous can be classified as a RKHS, with a single reproducing kernel associated with it (Steinwart & Christmann, 2008, Thr. 4.20). Secondly, every symmetric and positive-definite function $cov : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the reproducing kernel of a unique RKHS (Steinwart & Christmann, 2008)

Representing functions in a RKHS: The reproducing property is a key feature of an RKHS. It implies that evaluating a function in $\mathcal{H}$ at a point $\mathbf{x}$ can be done by taking the inner product of the function with the kernel evaluated at $\mathbf{x}$, which is computationally efficient and allowed for the use of kernel methods in machine learning and other fields.

Feature maps: Given an arbitrary Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$, mapping $\mathcal{X}$ into $\mathcal{H}$, we have that $cov(x, y) := \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}, \forall x, y \in \mathcal{X}$ defines a reproducing kernel for the Hilbert space $\mathcal{H}_{cov}$ of functions $f : \mathcal{X} \to \mathbb{R}$. Notice that the feature space $\mathcal{H}$ does not need to be $\mathcal{H}_k$ itself nor a function space. However, if $\mathcal{H} = \mathcal{H}_k$, ϕ is called the canonical feature map. In this case, it can be proved that ϕ is unique and given by $x \mapsto cov(x, \cdot)$ (Steinwart & Christmann, 2008).

Representing functions with finite features: For many practical problems, the optimal solution to a learning problem in a high or infinite-dimensional RKHS can be expressed as a finite linear combination of kernel functions evaluated at the training data points. This result was first discovered by Kimeldorf and Wahba (1970) and is known as the *representer theorem*.

Denote $\{(\mathbf{x}_i, y_i), i = 1, \cdots, N\}$ as N observations of a covariate $\mathbf{x} \in \mathcal{X}$ and a response $y \in \mathbb{R}$. Furthermore, assume our goal is to perform regression analysis to learn the function $f \in \mathcal{H}$, such that:

$$y_i = f(\mathbf{x}_i) + \eta_i, \quad i = 1, \cdots, N \tag{2.32}$$

where η_i are i.i.d. random errors with zero mean and $\mathcal{H}$ is a Hilbert space of function from $\mathcal{X} \to \mathbb{R}$. Since the space $\mathcal{H}$ is usually infinite dimensional, certain regularisation is necessary for estimation. Hence, we estimate f as the solution to the penalised least squares

$$\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - f(\mathbf{x}_i)\right)^2 + \lambda \|f\|_2 \tag{2.33}$$

where the first part measures the goodness-of-fit, and $\|f\|_2$ a regularisation function chosen without loss of generality. Let $\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}_1$ where $\mathcal{H}_0 = \{f : \|f\| = 0\}$ is a finite dimensional space with basis functions $\xi_1, \cdots, \xi_M$ and $\mathcal{H}_1$ is the RKHS induced by the kernel k_1. Let $\phi_i(\cdot) = cov(\mathbf{x}_i, \cdot)$ for $i = 1, \cdots, N$ be *representers*.

Theorem 1 (Representer theorem) *The solution to Eq.* (2.33)*, $\hat{f}$, is a linear combination of the basis functions $\xi_1, \cdots, \xi_M$ and representers $\phi_1, \cdots, \phi_N$:*

$$\hat{f} = \sum_{k=1}^{M} d_k \xi_k + \sum_{j=1}^{N} c_j \phi_j .$$

Proof Any $f \in \mathcal{H}$ can be expressed as $f = \sum_{k=1}^{M} d_k \xi_k + \sum_{j=1}^{N} c_j \phi_j + \rho$ where $\rho \in \mathcal{H}$ is orthogonal to the space spanned by $\xi_1, \cdots, \xi_M$ and $\phi_1, \cdots, \phi_N$. Then the penalised least squares Eq. (2.33) reduces to

$$\sum_{i=1}^{N} \left(y_i - \left\langle k_{\mathbf{x}_i}, \sum_{k=1}^{M} d_k \xi_k + \sum_{j=1}^{N} c_j \phi_j + \rho \right\rangle \right)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k_1(\mathbf{x}_i, \mathbf{x}_j) + \lambda \|\rho\|_2 \tag{2.34}$$

where $k_{\mathbf{x}_i} = cov(\mathbf{x}_i, \cdot)$ and $cov(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{M} \xi_k(\mathbf{x}) \xi_k(\mathbf{x}') + k_1(\mathbf{x}, \mathbf{x}')$ is the reproducing kernel of $\mathcal{H}$. Note that $cov(\mathbf{x}_i, \cdot)$ belongs to the subspace spanned by $\xi_1, \cdots, \xi_M$ and $\phi_1, \cdots, \phi_N$. Therefore, ρ drops out the first term in Eq. (2.34) since it is orthogonal to $cov(\mathbf{x}_i, \cdot)$. Consequently $\rho = 0$ and the conclusion follows.

2.6 Stein Variational Gradient Descent

One of the main applications of the theory of RKHS's in this work will be on kernel methods for VI, in particular on SVGD. Introduced by Liu and Wang, this algorithm serves as a natural counterpart of gradient descent for optimization, providing a general-purpose variational inference method.

The main challenge in VI arises in defining an appropriate $\mathcal{Q}$. SVGD addresses this issue while also solving for Eq. (2.16) by performing Bayesian inference in a nonparametric nature, removing the need for assumptions on restricted parametric families for $q(\mathbf{x})$ (Liu & Wang, 2016). We start from an uninformative proposal distribution and construct a empirical distribution through sampling a set of particles $\{\mathbf{x}_i\}_{i=1}^{N_p}$, $\mathbf{x} \in \mathbb{R}^p$. The method then reduces the KL divergence between our empirical and target distributions by computing incremental transforms to perform steepest descent in a RKHS. Unlike other methods, SVGD does not require explicit specification of parametric forms or calculation of the Jacobian matrix. Instead, it mimics the typical gradient descent algorithm, making it simple and intuitive.

More formally, the incremental transform in SVGD is defined by a small perturbation of the identity map

$$\mathrm{T}(\mathbf{x}) = \mathbf{x} + \epsilon\, \boldsymbol{\phi}(\mathbf{x}). \tag{2.35}$$

Here, $\boldsymbol{\phi}(\mathbf{x})$ is a smooth function that characterizes the perturbation direction, and ϵ is a scalar that represents the perturbation magnitude. Assuming we start from a simple reference distribution q_0, by the change of variables formula we get

$$q_{[\mathrm{T}]}(\mathbf{x}) = q\big(\mathrm{T}^{-1}(\mathbf{x})\big) \cdot \big|\det\big(\nabla_{\mathbf{x}}\, \mathrm{T}^{-1}(\mathbf{x})\big)\big|, \tag{2.36}$$

where $q_{[\mathrm{T}]}$ is the resulting distribution from applying the map T to the random vector $\mathbf{x}$. When $|\epsilon|$ is sufficiently small, the Jacobian is full-rank and close to the identity matrix, and T is guaranteed to be a one-to-one transform.

A key insight of SVGD is the connection it draws between the Stein operator and the derivative of the KL divergence. The derivative of KL divergence with respect to the perturbation magnitude ϵ is given by:

$$\nabla_\epsilon D_{\mathrm{KL}}(q_{[T]} \,||\, p)\Big|_{\epsilon=0} = -\, \mathbb{E}_{\mathbf{x}\sim q}[\mathrm{tr}(\mathcal{A}_p\, \boldsymbol{\phi}(\mathbf{x}))], \tag{2.37}$$

where $\mathcal{A}$ is the Stein operator. This equation forms the foundation of the SVGD method, providing a theoretical basis for the algorithm. The SVGD algorithm mimics a gradient dynamics updating each particle in parallel according to:

$$\mathbf{x}_i^{(l+1)} \leftarrow \mathbf{x}_i^{(l)} + \epsilon\, \boldsymbol{\phi}^*(\mathbf{x}_i^{(l)}). \tag{2.38}$$

The function $\boldsymbol{\phi}(\cdot)$ is known as the score function and defines the velocity field that maximally decreases the KL divergence:

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}\in\mathcal{H}}{\arg\max}\Big\{-\nabla_\epsilon D_{\mathrm{KL}}\big(q_{[\mathrm{T}]}(\mathbf{x}) \,||\, p(\mathbf{x})\big), \text{ s.t. } \|\boldsymbol{\phi}\|_{\mathcal{H}} \le 1\Big\}, \tag{2.39}$$

where $\mathcal{H}$ is a RKHS induced by a positive-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and $q_{[\mathrm{T}]}$ indicates the particle distribution resulting from taking an update step as in Eq. (2.38). Recall that an RKHS $\mathcal{H}$ associated with a kernel k is a Hilbert space of functions endowed with an inner product $\langle\cdot,\cdot\rangle$ such that $f(\mathbf{x}) = \langle f, k(\cdot,\mathbf{x})\rangle$ for any $f \in \mathcal{H}$ and any $\mathbf{x} \in \mathcal{X}$ (Schölkopf & Smola, 2002). The problem in (2.39) has been shown to yield a closed-form solution which can be interpreted as a functional gradient in $\mathcal{H}$ and approximated with the set of particles (Liu & Wang, 2016):

$$\boldsymbol{\phi}^*(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\hat{q}}\big[k(\mathbf{y},\mathbf{x})\,\nabla_{\mathbf{y}}\log p(\mathbf{y}) + \nabla_{\mathbf{y}}k(\mathbf{y},\mathbf{x})\big], \tag{2.40}$$

with $\hat{q} = \frac{1}{N_p}\sum_{i=1}^{N_p}\delta(\mathbf{x}^i)$ being an empirical distribution that approximates q with a set of Dirac delta functions $\delta(\mathbf{x}^i)$. For SVGD, k is typically a translation-invariant kernel, such as the squared-exponential or the Matérn kernels (Liu & Wang, 2016; Rasmussen & Williams, 2006). The two terms in Eq. (2.40) play different roles. The first term drives the particles towards the high probability areas of $p(\mathbf{x})$ by following a smoothed gradient direction, which is the weighted sum of the gradients of all the points weighted by the kernel function. The second term acts as a repulsive force that prevents all the points from collapsing together into local modes of $p(\mathbf{x})$. The method is summarised in Algorithm 1.

Algorithm 1: Stein Variational Gradient Descent

```
1 Input: A target distribution with density function $p(\mathbf{x})$ and a set of initial
    particles $\{\mathbf{x}_i^0\}_{i=1}^{N_p}$.
2 Output: A set of particles $\{\mathbf{x}_i\}_{i=1}^{N_p}$ that approximates the target distribution.
3 for each iteration $l$ do
4     $\boldsymbol{\phi}^*(\mathbf{x}) = \frac{1}{N_p}\sum_{j=1}^{N_p}[k(\mathbf{x}_j,\mathbf{x})\,\nabla\log p(\mathbf{x}_j) + \nabla\, k(\mathbf{x}_j,\mathbf{x})]$ ;
5     $\mathbf{x}_i^{(l+1)} \leftarrow \mathbf{x}_i^{(l)} + \epsilon\,\boldsymbol{\phi}^*(\mathbf{x}_i^{(l)})$;
6 end
```

The SVGD algorithm, with its simplicity and efficiency, provides a powerful tool for Bayesian inference. By leveraging the connection between the Stein operator and the derivative of KL divergence, SVGD offers a novel approach to variational inference.

2.7 Optimal Control

In classical control the design of control systems generally involve the use of transfer functions, which describe the relationship between the input and output of a known system in the frequency domain, to analyse and determine the design parameters of an *acceptable* control system. Acceptable performance is generally defined in terms of time and frequency domain criteria such as rise time, settling time, peak overshoot, gain and phase margin, and bandwidth. However, for many real-world problems this involves satisfying several different performance criteria for complex systems with multiple-input and multiple-output. For example, the design of a spacecraft attitude control system that minimises fuel expenditure is not easily solved using classical control (Kirk, 2012).

Optimal control offers a more direct approach and became the de-facto standard for control systems. The objective of optimal control theory is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimise (or maximise) some performance criterion. The basic approach involves formulating a mathematical model of the dynamic system, along with a cost function that captures the desired objective. The model typically consists of a set of differential equations that describe how the state of the system changes over time, and the control input is a function that drives the state to the desired trajectory. On the other hand, a cost functional is used as a surrogate to represent some measure of the system's performance.

The goal is to find the control input that minimises the cost function, subject to the system dynamics, and any other constraints. Consider the problem of controlling a discrete-time system described by a non-linear set of difference equations of the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{2.41}$$

where f is the transition function, $\mathbf{x}_t \in \mathbb{R}^n$ denotes the system states, $\mathbf{u}_t \in \mathbb{R}^m$ are the control inputs at a given time t. The optimal control problem can then be posed as an optimisation problem:

$$\begin{aligned} \mathbf{u}^* \coloneqq \arg\min_{\mathbf{u}} \quad & \mathcal{C}(f, \mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \\ & \mathcal{H}(\mathbf{x}_t, \mathbf{u}_t) = 0 \\ & \mathcal{G}(\mathbf{x}_t, \mathbf{u}_t) \leq 0 \\ & \text{for } t = 0, 1, 2, \ldots \end{aligned} \tag{2.42}$$

where $\mathcal{C}$ is the cost functional, and $\mathcal{H}$ and $\mathcal{G}$ are a set of equality and inequality constraints, respectively.

For linear system or well-behaved non-linear system where the system dynamics are known, optimal controllers can be designed using Pontryagin's maximum principle or the Hamilton-Jabobi-Bellman equation (HJB) offline, although for most practical cases such solutions are unfeasible Lewis, Vrabie and Vamvoudakis, 2012; Ross, 2009. Furthermore, even though HJB can be extended to stochastic systems, such as in the case of unknown disturbances, it assumes that distributions are stationary over time. Additionally, in practice there are many problems in which the dynamics f are unknown, or may be too complex to represent analytically. As a result, it is necessary to employ numerical methods to solve optimal control problems over a finite look ahead horizon H.

2.8 Model Predictive Control

MPC is the most commonly used is a type of optimal control to circumvent the issues above, and in particular for real-time control of dynamic (Camacho & Alba, 2013). MPC involves solving a finite-horizon optimal control problem at each time step, subject to constraints on the state and control variables, and using only the first control input from the optimal control sequence to control the system. The process is then repeated at the next time step, with the time horizon shifted forward by one step. This process is known as RHC, which means that the control inputs are updated at each time step based on the current state of the system and the predicted future evolution of the system. The RHC formulation allows MPC to account for uncertainties and disturbances in the system, as well as time-varying constraints on the state and control variables. MPC is widely used in a variety of applications, including process control, robotics, agriculture, and automotive control (Ding et al., 2018; Erez et al., 2013; Mayne, 2014). It is particularly well-suited for systems with constraints or uncertainties, and has been shown to be effective in achieving good control performance and robustness in practice (Mesbah, 2016).

Therefore, rather than optimising over a deterministic control sequence, our goal is now to optimise over *policies*. A policy is a function π that maps a desired system trajectory to a sequence of control actions $\mathbf{u}$. Therefore we can modify our formulation in Eq. (2.42) where the policy π is updated at every time step based only on the previous trajectory.

$$\begin{aligned} \pi^* \coloneqq \arg\min_{\pi} \quad & \mathbb{E}_{\pi}[\mathcal{C}(f, \mathbf{x}, \mathbf{u})] \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \\ & \mathbf{u}_t = \pi(\mathbf{x}_t) \\ & \mathcal{H}(\mathbf{x}_t, \mathbf{u}_t) = 0 \\ & \mathcal{G}(\mathbf{x}_t, \mathbf{u}_t) \leq 0 \\ & \text{for } t = 0, 1, \dots, H, \end{aligned} \tag{2.43}$$

for some initial state $\mathbf{x}_0$.

There are several methods for solving optimal control problems, including dynamic programming, calculus of variations, numerical optimisation, and, more recently, approximate inference (Watson, Abdulsamad & Peters, 2020). Constraints in Eq. (2.43) may include physical limitations on the control input, such as maximum torque or voltage, or limitations on the state variables, such as staying within a certain range of values. Indeed, one of the key challenges is balancing the competing objectives of minimising the cost function while also satisfying the system constraints. Another important consideration in optimal control is the trade-off between the accuracy of the mathematical model and the computational cost of solving the optimisation problem. More complex models may capture the system dynamics more accurately, but are more expensive to solve. Simplified models may be more computationally efficient, but may not capture all the important features of the system.

Note how in this formulation, MPC is similar to a RL problem. The main difference being that in contemporary RL the main paradigm is what known as *Episodic Reinforcement Learning* Sutton and Barto, 2018. Under this paradigm, our policy π is typically learned offline and updated only after a complete episode. Our goal is to find a policy that maximises the reward, i.e. minimises the cost $\mathcal{C}$, in the least amount of episodes.

2.9 Stochastic Non-linear Model Predictive Control

Stochastic Non-linear Model Predictive Control (SNMPC) is an extension of deterministic MPC that takes into account uncertainties in the system dynamics or disturbances that affect the system behavior. In SNMPC, the system is modeled as a stochastic process, and the control inputs are chosen to optimise a probabilistic measure of performance, such as the expected value or variance of the cost function.

In other words, we are interested in the problem where the real transition function $f(\mathbf{x}_t, \mathbf{u}_t)$ is unknown and approximated by a transition function with *parametric uncertainty*, such that Eq. (2.41) can be rewritten as:

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \hat{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\xi}) := f_{\boldsymbol{\xi}}(\mathbf{x}_t, \mathbf{u}_t) \tag{2.44}$$

where $\boldsymbol{\xi} \in \varXi$ are the parameters of the model with a prior probability distribution $p(\boldsymbol{\xi})$ and the non-linear forward model $\hat{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\xi})$, is represented as $f_{\boldsymbol{\xi}}$ for compactness.

The main difference to our original problem formulation is that we now need to take into account the model uncertainty when computing statistical measures of our system. Namely, we can modify Eq. (2.43) to its stochastic counterpart:

$$\begin{aligned} \arg\min_{\pi} \quad & \hat{J} := \mathbb{E}_{\pi,\boldsymbol{\xi}}[\mathcal{C}(f_{\boldsymbol{\xi}}, \mathbf{x}, \mathbf{u})] \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = f_{\boldsymbol{\xi}}(\mathbf{x}_t, \mathbf{u}_t) \\ & \mathbf{u}_t = \pi(\mathbf{x}_t) \\ & \mathcal{H}(\mathbf{x}_t, \mathbf{u}_t) = 0 \\ & \mathcal{G}(\mathbf{x}_t, \mathbf{u}_t) \leq 0 \\ & \text{for } t = 0, 1, \dots, H, \end{aligned} \tag{2.45}$$

where we define the estimator $\hat{J}$ and $\mathbf{x}_0$ is once more some predetermined initial state.

The main strategies used to solve MPC can in broad part be utilised for SNMPC, although sampling-based solutions and approximate inference are more commonly used as well. One of the main advantages of sampling-based SNMPC is that $\hat{J}$ can be computed even when the cost function is non-differentiable w.r.t. to the policy parameters by using importance sampling (G. Williams, Aldrich & Theodorou, 2017). In the next section we present one sampling based SNMPC method which will be used extensively as a baseline throughout this thesis.

2.10 Model Predictive Path Integral

One of the biggest barriers of SNMPC is the solution of complex, non-linear, non-convex optimisation problems. A possible approach to make such problems tractable is to resort to sampling-based, also known as shooting, control methods. Recently, G. Williams et al. (2016) presented a sampling-based MPC approach grounded on an information theoretic approach to Stochastic Optimal Control problems denominated MPPI—also known as Information Theoretical MPC (IT-MPC)—which we outline below. MPPI will provide the basis upon which the method in Chapter 3 is developed and will be used extensively as a baseline throughout this dissertation.

Consider a discrete-time stochastic system described by a non-linear set of difference equations of the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{v}_t) \tag{2.46}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ denotes the system states at time t, $\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}, \boldsymbol{\Sigma}) \in \mathbb{R}^m$ is the system input at a given time t and f denotes a typically non-linear discrete transition function.

For all purposes, in this section we will assume that f is time-invariant. Furthermore, we assume that a finite time-horizon H and control frequency are given, which uniquely defines the units of time of the system. A final assumption is that we do not have direct control over the variable $\mathbf{v}$, but we are able to control its mean $\mathbf{u}$. In other words, $\mathbf{v}$ is a random vector generated by a white-noise process with the following density function:

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \boldsymbol{\Sigma}) \in \mathbb{R}^{m\times H}, \tag{2.47}$$

where we are free to set $\mathbf{u}_t$ at any given discrete-time.

Hence, we can proceed by defining a fixed length input sequence

$$U = (\mathbf{u}_0, \dots, \mathbf{u}_{H-1}) \tag{2.48}$$

over a fixed control horizon H, onto which we apply a RHC strategy. This yields

$$V = (\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{H-1}) \in \mathbb{R}^{m\times H}, \tag{2.49}$$

which is itself a random variable.

Furthermore, let's denote as $p(V)$ the Probability Density Function (PDF) of the uncontrolled system (i.e. $U \equiv \mathbf{0}$) and, similarly, let's denote by $q(V)$ the corresponding PDF for the open-loop control sequence (that is whenever $U \neq \mathbf{0}$). The optimal control problem may then be defined as:

$$U^* = \underset{U \in \mathcal{U}}{\arg\min}\, \mathbb{E}_q\left[\mathcal{C}_{\text{term}}(\mathbf{x}_H) + \sum_{t=0}^{H-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) \right], \tag{2.50}$$

where $\mathcal{U}$ is the set of admissible controls, $\mathcal{C}_{\text{term}}(\mathbf{x}_H)$ is a terminal cost function, and $\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t)$ is a running cost function of the form:

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) = \mathcal{C}_{\text{inst}}(\mathbf{x}_t) + \frac{\alpha}{2}\left(\mathbf{u}_t^\intercal \mathbf{\Sigma}^{-1} \mathbf{u}_t + \tilde{\mathbf{u}}_t^\intercal \mathbf{u}_t\right), \tag{2.51}$$

where $\alpha \in \mathbb{R}^+$ is known as the inverse temperature and the affine term $\tilde{\mathbf{u}}$ allows the vector of minimum control (rest position) to be different from zero. Noting that the state cost may be considered independent from the control terms, we can define

$$\mathcal{C}(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_H) = \mathcal{C}_{\text{term}}(\mathbf{x}_H) + \sum_{t=0}^{H-1} \mathcal{C}_{\text{inst}}(\mathbf{x}_t). \tag{2.52}$$

Moreover, we define a functional mapping, H, from input sequences V to their resulting trajectory by recursively applying f given $\mathbf{x}_0$. That is,

$$\mathrm{H}(V \mid \mathbf{x}_0) = \left[\mathbf{x}_0, f(\mathbf{x}_0, \mathbf{v}_0), f(f(\mathbf{x}_0, \mathbf{v}_0), \mathbf{v}_1), \dots\right]. \tag{2.53}$$

The final state-cost function can then be defined by the following functional composition:

$$\mathcal{S}(V \mid \mathbf{x}_0) = \mathcal{C}\big(\mathrm{H}(V \mid \mathbf{x}_0)\big), \tag{2.54}$$

where for simplicity the dependency on the initial state $\mathbf{x}_0$ will be dropped and the state-cost simply denoted by $\mathcal{S}(V)$.

Finally, IT-MPC relies on the *free-energy* principle to compute a lower bound for the optimal control problem. The free-energy of a control system is given by

$$\mathrm{F}(\mathcal{S}, p, \mathbf{x}_0, \alpha) = -\alpha \log\left(\mathbb{E}_p\left[\exp\left(-\frac{1}{\alpha}\, \mathcal{S}(V)\right)\right]\right), \tag{2.55}$$

and roughly defines the bounds the information gained when sampling a generative model. Here again α is the inverse temperature and p is a base probability density akin to a Bayesian prior.

It follows that Eq. (2.55) defines the form of the optimal distribution function $q^*(V)$ for which the bound on the optimal control problem is tight, thus achieving the optimal control U^* (G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018):

$$q^*(V) = \frac{1}{\eta} \exp\Big(-\frac{1}{\alpha}\, \mathcal{S}(V)\Big)\, p(V) \tag{2.56}$$

$$\eta = \int \exp\Big(-\frac{1}{\alpha}\, \mathcal{S}(V)\Big)\, p(V)\, \mathrm{d}V, \tag{2.57}$$

where η^* is the normalising term spanning $\mathbb{R}^{m \times H}$. This results in

$$\mathbf{u}_t^* = \mathbb{E}_{q^*}[\mathbf{v}_t], \;\; \forall t \in \{0, 1, \dots H-1\}. \tag{2.58}$$

Therefore, the optimal open-loop control sequence is the expected value of control trajectories sampled from the optimal density. As we cannot sample directly from q^*, we can resort to importance sampling (Andrieu et al., 2003) to construct an unbiased estimator of the optimal distribution, given the current control distribution, namely

$$\mathbb{E}_{q^*}[\mathbf{v}_t] = \int q^*(V)\mathbf{v}_t\, \mathrm{d}V = \int \omega(V)\, q(V \mid \hat{U}, \mathbf{\Sigma})\mathbf{v}_t\, \mathrm{d}V, \tag{2.59}$$

where $\omega(V) = q^*(V)/\, q(V \mid \hat{U}, \mathbf{\Sigma})$ is the importance sampling weight and $\hat{U}$ the estimated control sequence.

In practice it may well be the case that the base distribution for the system at rest is not centred at zero. In other words, the baseline value for $\mathbf{u}_t$ at any given time is not the usual zero vector $\mathbf{0}$. This can be accounted for by the inclusion of a minimum control value, such that $\mathbf{u}_t = (\hat{\mathbf{u}}_t - \tilde{\mathbf{u}}_t)$ is the difference between the current control action $\hat{\mathbf{u}}_t$ and the minimum control $\tilde{\mathbf{u}}_t$.

Finally, we can switch the expectation to $\mathbb{E}_{q(\hat{U})}$, resulting in $\mathbb{E}_{q(\hat{U})}[\omega(V)\mathbf{v}_t]$. We can then use the definition of the optimal distribution w.r.t. the base measure distribution (G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018) to derive the *optimal* *information-theoretic control law*:

$$\omega(V) = \frac{1}{\eta}\exp\left(-\frac{1}{\alpha}\left(\mathcal{S}(V) + \alpha\sum_{t=0}^{H-1}\mathbf{u}_t^\top\boldsymbol{\Sigma}^{-1}\mathbf{v}_t\right)\right) \tag{2.60}$$

$$\mathbf{u}_t = \mathbb{E}_{q(\hat{U})}\left[\omega(V)\mathbf{v}_t\right], \tag{2.61}$$

where:

$$\eta = \int\exp\left(-\frac{1}{\alpha}\left(\mathcal{S}(V) + \alpha\sum_{t=0}^{H-1}\hat{\mathbf{u}}_t^\top\boldsymbol{\Sigma}^{-1}\mathbf{v}_t\right)\right)\mathrm{d}V. \tag{2.62}$$

Note that for numerical stability it is customary to multiply the numerator and denominator of $\omega(V)$ by a factor $\exp\left({}^1\!/\!_\alpha\tilde{C}\right)$, where $\tilde{C}$ is defined as the minimum cost.

2.11 Sim-to-real

Sim-to-real is the process of transferring a control policy learned in a simulated environment to a real-world environment. This process is challenging due to the differences between the simulation and the real world, such as different dynamics, sensory inputs, and environmental factors. Collectively, these differences account for a reduced performance when an agent trained in simulation is deployed in the real-world, commonly referred to as the *reality gap*. Reducing the reality gap is the key challenge in sim-to-real. In recent years there has been a growing interest in developing techniques that can improve the sim-to-real transfer, and several approaches have been proposed.

In domain randomisation the main premise is to randomise the parameters of the simulation to make it more diverse and robust to variations in the real world. To generate a large number of training examples in simulation, various aspects of the environment are randomly modified, such as lighting, object textures, camera angles, and physics parameters. By varying these parameters in a controlled way, the model can learn to generalise to a wide range of real-world scenarios (Peng et al., 2018). For example, a robot trained with domain randomisation might learn to grasp objects in a variety of lighting conditions, or navigate through environments with different textures and obstacles. The main advantage of domain randomisation is that it allows policies to be trained with a much larger and more diverse set of examples than would be possible in the real world, usually at a fraction of the cost. This can help to improve the robustness and generalisation of the model when deployed in the real world, where the environment is often unpredictable and complex. However, one of the main challenges is to strike a balance between generating enough variation in the simulation environment to improve generalisation, while also ensuring that the training examples are still relevant to the real-world scenario. Furthermore, Valassakis, Ding and Johns (2020) found effectively no difference in training robust policies, i.e. resilient to certain perturbations, and domain randomisation.

Another popular approach is known as transfer learning. This involves using pretrained policies in simulation as a starting point for training in the real world. The assumption is that by initialising the agent with knowledge from simulation, it will be able to learn faster and more effectively in the real world (Heess et al., 2017). This approach can be especially useful in situations where it is difficult or expensive to obtain real-world data but it requires a simulated environment that closely resembles the real-world environment. This can involve building a detailed simulation model that includes all of the relevant physics, sensors, and other environmental factors that the robot will encounter in the real world.

Finally, in curriculum learning the strategy used is to train a policy or model in simulation using a curriculum, i.e. a series of gradually increase difficulty tasks. This approach is motivated by the fact that many real-world problems are complex and cannot be solved using a single model or algorithm (Andrychowicz et al., 2020) In the context of robotic control, for example, curriculum learning can be used to train a robot to perform a specific task, such as grasping an object or navigating through an environment. The idea is to start with simple tasks that the robot can easily perform and gradually increase the difficulty of the tasks as the robot becomes more proficient. For example, to train a robot to grasp an object, the curriculum might start with simple objects that are easy to grasp, such as a ball or a cube. Once the robot has mastered these simple objects, the curriculum can gradually increase the difficulty of the objects, such as objects with irregular shapes or objects that are more difficult to grip. The underlying assumption of introducing a curriculum is that it will induce a underlying structure to the learned policy, such as motion primitives for grasping, that will be useful in generalising to harder tasks or environments. This approach can also help to avoid the problem of the robot getting stuck in a local minimum, which

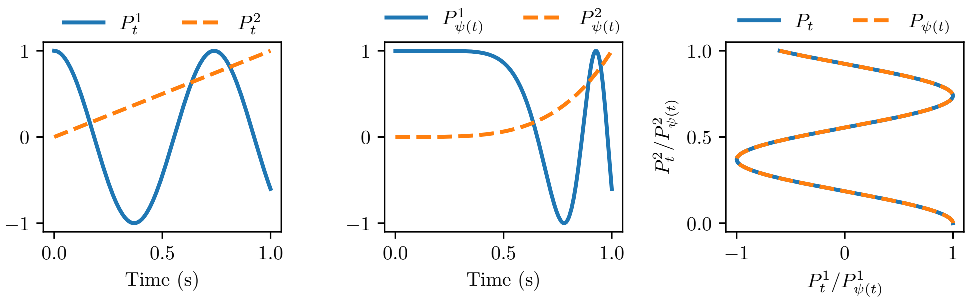


Figure 2.1: **Signature invariance to reparametrisation.** *Left*: Plot of the coordinates of a two dimensional path P_t over time. Here $P_t^1 = \cos(8.5t)$ and $P_t^2 = t$. *Centre*: Plot of the two coordinates of path P_t reparameterised by function ψ. Now, $P_{\psi(t)}^1 = \cos(8.5t^4)$ and $P_{\psi(t)}^2 = t^4$. *Right*: Plots of path P_t and its reparameterised version $P_{\psi(t)}$ are shown overlapping to illustrate how the change in time is irrelevant if the goal is achieving diverse paths. The signature of degree 2 for both paths is $\{1, -1.6, 1, 1.3, -0.9, -0.7, 0.5\}$.

can occur when the robot is trained on a difficult task from the beginning.

2.12 Path signature

A multitude of practical data streams and time series can be regarded as a path, for example, video, sound, financial data, control signals, handwriting, etc. The path signature transforms such multivariate sequential data (which may have missing or irregularly sampled values) into an infinite-length series of real numbers that uniquely represents a trajectory through Euclidean space. Although formally distinct and with notably different properties, one useful intuition is to think of the signature of a path as akin to a Fourier transform, where paths are summarised by an infinite series of feature space coefficients. Consider a path X traversing space $\mathcal{X} \subseteq \mathbb{R}^c$ as defined in Section 2.8. Note that at any time t such path can be decomposed in

$$X_t = \{X_t^1, X_t^2, \dots, X_t^c\}. \tag{2.63}$$

Now recall that for a one-dimensional path X_t and a function f, the path integral of f along X is defined by:

$$\int_a^b f(X_t)\,\mathrm{d}X_t = \int_a^b f(X_t)\dot{X}_t\,\mathrm{d}t. \tag{2.64}$$

In particular, note that the mapping $t \to f(X_t)$ is also a path. In fact, Eq. (2.64) is an instance of the Riemann-Stieltjes integral (Chevyrev & Kormilitzin, 2016):

$$\int_a^b Y_t \, \mathrm{d}X_t = \int_a^b Y_t \dot{X}_t \, \mathrm{d}t, \tag{2.65}$$

which computes the integral of one path against another. Let us now define the *1-fold iterated* integral, which computes the increment of the i-th coordinate of the path at time t as:

$$\mathrm{S}(X)^i_t = \int\limits_{a<t_1<t} \mathrm{d}X^i_{t_1} = X^i_t - X^i_a, \tag{2.66}$$

and we again emphasise that $\mathrm{S}(X)^i_t$ is also a real valued path. This allows us to apply the same iterated integral recursively and we proceed by defining the *2-fold iterated* integral (Chen, 1954, 1977) as:

$$\mathrm{S}(X)^{i,j}_t = \int\limits_{a<t_2<t} \mathrm{S}(X)^i_{t_2} \, \mathrm{d}X^j_{t_2} = \int\limits_{a<t_1<t_2<t} \mathrm{d}X^i_{t_1} \, \mathrm{d}X^j_{t_2}. \tag{2.67}$$

Informally, we can proceed indefinitely and we retrieve the path signature by collecting all iterated integrals of the path X. A geometric intuition of the signature can be found in (Chevyrev & Kormilitzin, 2016; Yang et al., 2017) where the first three iterated integrals represent displacement, the Lévy area (T. J. Lyons, Picard & Lévy, 2007) and volume of the path respectively.

Definition 9 (Signature) The *signature* of a path $X : t \in [a, b] \to \mathbb{R}^c$, denoted by $\mathrm{S}(X)_t$, is the infinite series of all iterated integrals of X. Formally, $\mathrm{S}(X)_t$ is the sequence of real numbers

$$\mathrm{S}(X)_t = \left(1, \mathrm{S}(X)^1_t, \dots, \mathrm{S}(X)^c_t, \mathrm{S}(X)^{1,1}_t, \mathrm{S}(X)^{1,2}_t, \dots\right), \tag{2.68}$$

where the iterated integrals are defined as:

$$\mathrm{S}(X)^{i_1,\dots,i_k}_t = \int\limits_{a<t_k<t} \dots \int\limits_{a<t_1<t_2} \mathrm{d}X^{i_1}_{t_1} \dots \mathrm{d}X^{i_k}_{t_k}, \tag{2.69}$$

and the superscripts are drawn from the set $\mathcal{M}$ of all multi-indexes,

$$\mathcal{M} = \left\{(i_1, \dots, i_k) \mid k \geq 1, i_1, \dots, i_k \in \{1, \dots, c\}\right\}. \tag{2.70}$$

In practice we often apply a truncated signature up to a degree d, that is $\mathrm{S}^d(X)_t$, defined as the finite collection of all terms of the signature up to multi-indices of length d.

The path signature was originally introduced by Chen (1958) who applied it to piecewise smooth paths and further developed by Lyons and others (Améndola, Friz & Sturmfels, 2019; Boedihardjo et al., 2016; Hambly & Lyons, 2010; T. Lyons, 2014). The number of elements in the path signature depends on the dimension of the input c and the degree d, and is given by c^d. Therefore the time and space scalability of the signature is rather poor ($O(c^d)$), but this can be alleviated with the use of kernel methods as we will discuss in Section 5.3. The signature of a path has several interesting properties which make it inherently interesting for applications in robotics.

Canonical feature map for paths For all effects, the path signature can be thought of as a *linear* feature map (Fermanian, 2021) that transforms multivariate sequential data into an infinite length series of real numbers which uniquely represents a trajectory through Euclidean space. This is valid even for paths with missing or irregularly sampled values (Boedihardjo et al., 2016; Hambly & Lyons, 2010).

Time-reversal We informally define the time-reversed path $\overleftarrow{X}$ as the original path X moving backwards in time. It follows that the tensor product of the signatures $\mathrm{S}(X)_{a,b} \otimes \mathrm{S}(\overleftarrow{X})_{a,b} = 1$, which is the identity operation.

Uniqueness The signature of every non tree-like path is unique (Hambly & Lyons, 2010). A tree-like path is one in which a section exactly retraces itself. Tree-like paths are quite common in real data (e.g. in cyclic actions) and this could be a limiting factor of the signature's application. However, it has been proven (Hambly & Lyons, 2010) that if a path has at least one monotonous coordinate, then its signature is unique. The main significance of this result is that it provides a practical procedure to guarantee signature uniqueness by, for example, including a time dimension.

Invariance under reparametrisation An important difficulty when vying for diversity in trajectory optimisation is the potential symmetry present in the data. This is particularly true when dealing with sequential data, such as, for instance, trajectories of an autonomous vehicle. In this case, the problem is compounded as there is an infinite group of symmetries given by the reparametrisation of a path (i.e. continuous surjections in the time domain to itself), each leading to distinct similarity metrics. In contrast, the path signature acts as a filter that is invariant to reparametrisation removing these troublesome symmetries and resulting in the same features as shown in Figure 2.1.

Dimension is independent of path length The final property we will emphasise is how the dimension of the signature depends on its degree and the intrinsic dimension of the path, but is independent of the path length. In other words, the signature dimension is invariant to the degree of discretisation of the path.

All models are wrong, but some are useful.

George Box

CHAPTER 3

Propagating model uncertainty in MPC through moment matching

Robustness to model miss-specification and noisy sensor measurements is a critical property for control systems operating in complex robotics applications. To address such requirements many control strategies frameworks have been proposed of which model predictive control is one of the most successful and popular (Camacho & Alba, 2013). MPC has become a primary control method for handling non-linear system dynamics and constraints on input, output and state, taking into account performance criteria. It originally gained popularity in chemical and processes control (Eaton & Rawlings, 1991), being more recently adapted to various fields, such agricultural machinery (Ding et al., 2018), automotive systems (Di Cairano & Kolmanovsky, 2019), and robotics (Kamel et al., 2017; Zanon et al., 2014). In MPC one seeks to iteratively find the solution of an optimisation problem for a receding finite time-horizon using an approximate model of the system.

The development of powerful and more realistic simulators allows practitioners to analyse and verify the performance of the controller against these variables before the controller is deployed to the real robot. However, when the dynamic model is given by complex simulators that incorporate differential equations and numerical solvers there is little hope the equations can be reversed to reason about the parameters of the simulation to best match the real behaviour of the system. Furthermore, the simulator might abstract away the equations and solver from the user. Effectively, it can be interpreted as a generative model that can be sampled from given a set of parameter values, but not inverted. In this chapter we pose the question, can we leverage the power of simulators, treated as generative models, to design control strategies that are robust to parameter uncertainty?

On the other hand, the application of MPC to linear systems has been an active research area for many decades with extensive deployments to many practical problems (Barcelos & Camponogara, 2010; Mayne, 2014; Mesbah, 2016). Notably, the most common setting for linear MPC application are tasks that involve trajectory tracking or stabilisation. However, control tasks in reinforcement learning are usually more complex and therefore less suitable to linearisation, motivating the use of non-linear models (Recht, 2018). Another motivation for more complex models is the ability to use of more expressive constraints, even if not directly involved in the physical process, such as economic criteria (Bradford & Imsland, 2018). Despite its vast application in the linear case, the use of MPC in non-linear systems continues to be an increasingly active area of research in control theory (Mayne, 2014; Mesbah, 2016).

Recent work in the field has led to controllers that are able to incorporate non-linear dynamics without relying on linear or quadratic approximations (G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018; G. Williams et al., 2017). However, most MPC controllers still do not consider uncertainty in the parameters of their internal simulator for future trajectories. In addition, estimating parameters for the system's model usually requires large amounts of data from the real system, which can be infeasible for some applications. Yet, whenever the stochastic system uncertainties can be adequately modelled, it is more natural to explicitly take them into account in the control design method itself. In Stochastic Model Predictive Control (SMPC), the uncertainty on the internal system dynamics is intrinsic to the optimal control problem solved at every time step. This allows the controller to trade-off performance and satisfaction of the constraints by regulating the joint probability distribution of the system states and outputs (Mesbah, 2016).

In this chapter we make the following contributions: we develop a Stochastic Non-linear MPC variant which leverages recent advancements in likelihood-free inference to estimate both the uncertainty on the simulator parameters as well as to propagate it throughout the estimated trajectories. We call our method Double Likelihood-free Inference Stochastic Control (DISCO). The posterior distribution for the parameters of the simulator is estimated by combining simulated data from generative models and observations from the physical system. Using this posterior distribution allows us to take into account the uncertainty about the system's dynamics in the decision making process during the control task. We proceed to show that the UT (Julier & Uhlmann, 2004) provides a computationally efficient alternative when compared to traditional Monte Carlo approaches to propagate the uncertainty from the parameter space to the forward modelling of the trajectory rollouts. In short, Double Likelihood-free Inference Stochastic Control (DISCO) can be seen as a variant of the MPPI control algorithm (G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018) that considers the uncertainty in the system's parameters in its internal trajectory simulations.

3.1 Related work

The use of MPC in the control of linear systems is mature and has been widely studied and applied to real systems. However, as seen in (Mayne, 2014), Non-linear Model Predictive Control (NMPC) is still an open-research question, especially for systems were uncertainty over parameters and constraints on controls and state-space are considered. The most common methods for controlling general non-linear systems are based on Non-linear Programming (NLP) (Houska, Ferreau & Diehl, 2011) and Differential Dynamic Programming (DDP) (Erez et al., 2013). Both rely on approximations of dynamics and cost functions so that the online optimisation problem becomes tractable. However, these mainstream gradient-based MPC approaches have some shortcomings. In the DDP method, the cost function must be smooth and it is notoriously difficult to include state constraints. Whereas with NLP, constraints may be easily accounted for but a common issue is to define what to do when no feasible solution is found.

In Mesbah, 2016, a family of SNMPC methods are discussed. In Tube-based NMPC the objective of the control policy is to ensure that the forward trajectories will remain inside a desirable tube centred around a given trajectory, however the boundary tube has to be computed offline (Rakovic et al., 2012). A multi-stage NMPC approach has been suggested in which the uncertainty is modelled by a scenario tree approach from stochastic programming. However, the procedure quickly becomes intractable, since the size of the optimisation problem scales exponentially with the time horizon, number of uncertainties and uncertainty levels (Thangavel et al., 2017).

In Dhar and Bhasin, 2018, a bounded and asymptotically convergent adaptive law to refine parameter estimation is presented, but only for a linear MPC formulation. In contrast, the method in (Sakhdari & Azad, 2018) decouples the transition model for a Tube NMPC into a nominal model to impose system constraints and an adaptive model used for the control optimisation. However, the adaptive model uses only point-base estimates computed with least squares.

Although many of the methods above focus on robustness, they do not incorporate uncertainty over the parameters of the transition function. In Bradford and Imsland, 2018, this is accounted for by using a SNMPC with an Unscented Kalman Filter to propagate the uncertainty over the state-space. However, this method requires an optimisation with chance constraints to be solved online and, to keep the problem feasible, the variance of the trajectories has to be artificially constrained. The most similar approach is perhaps presented in (Arruda et al., 2017), where the MPPI formulation is used in conjunction with a Ensemble of Mixture Density Networks (E-MDN) to approximate the joint probability distribution of states and actions. This is similar to our approach, however as the E-MDN tries to approximate the joint distribution of states and actions, it needs to be retrained entirely on new environments.

In contrast, the variant of MPPI proposed in this chapter uses the UT to propagate the uncertainty over model parameters. This reduces the dimensionality of the inference problem and results in a controller more adept to generalise to unseen situations. Moreover, unlike the stochastic optimisation strategies, our framework is also amenable to the inclusion of constraints, as the control update law is based on sampled trajectories. As shown in (G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018) constraints may be applied directly to the control actions. On the other hand, we can apply soft constraints to the state space through the cost function. This is easily achieved as there is no need for the cost function to be differentiable and assures that a feasible solution will exist.

Additionally, DISCO takes advantage of the BayesSim LFI framework presented in (Ramos, Possas & Fox, 2019) to update the model uncertainty periodically. Hence, given a set of true observations after a specified episode length, we can update our knowledge of the posterior probability density of parameters $p(\boldsymbol{\xi} \mid \mathbf{x} = \mathbf{x}^r)$. This way our model can adapt to variations in the environment, e.g. adjust friction coefficients in case of rain, or intrinsic to the transition function, e.g. change of weight distribution. In contrast to other inference methods, such as Variational Inference or Markov Chain Monte Carlo, where a likelihood function is needed, in LFI we compute an approximated parametric distribution of the true posterior. Furthermore, BayesSim was shown to be more data efficient than other LFI methods, such as Approximate Bayesian Computation (Ramos, Possas & Fox, 2019).

3.2 Preliminaries

Once more, we consider the problem of controlling a discrete-time stochastic system described by a non-linear set of difference equations of the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{v}_t) \tag{3.1}$$

where f is the transition function, $\mathbf{x}_t \in \mathbb{R}^n$ denotes the system states, and $\mathbf{v}_t \in \mathbb{R}^m$ is the control input at a given time t. Note that there is no direct control over the variable $\mathbf{v}$, but we are able to control its mean $\mathbf{u}$. In other words, $\mathbf{v} \sim \mathcal{N}(\mathbf{u}, \boldsymbol{\Sigma})$ is the resulting change of an applied control signal with the addition of an stochastic component. This assumption is akin to a multiplicative noise model which is common in robotics, where lower-level actuator controllers are usually present, but also takes into account a bespoke amount of exploration in our control actions. As such, in practice, $\boldsymbol{\Sigma}$ is a hyper-parameter of our control system that may need to be adjusted. Finally, we assume a finite time-horizon H, and that the control frequency is given.

More generally, we are interested in the problem where the real transition function $f(\mathbf{x}_t, \mathbf{v}_t)$ is approximated by a parameterised non-linear forward model $f(\mathbf{x}_t, \mathbf{v}_t \mid \boldsymbol{\xi})$. In what follows, this approximated model will be represented as $f_{\boldsymbol{\xi}}$ for compactness and we may rewrite Eq. (3.1) as:

$$\mathbf{x}_{t+1} = f_{\boldsymbol{\xi}}(\mathbf{x}_t, \mathbf{v}_t). \tag{3.2}$$

This formulation is extremely useful in practice as we usually do not have access to the real transition function or, even if we do, in many cases we may want to use a simplified version of the system dynamics for performance reasons. Therefore, Eq. (3.2) provides not only an approximated version of Eq. (3.1), but one that can be modified and updated through a fixed set of parameters $\boldsymbol{\xi}$. As we will see throughout the chapter, this parametric approach will allow us to reason over model uncertainty and its effects on the control policy.

As seen in Section 2.8, at its core, model-based control relies on an approximated transition function to optimise the control actions over a given control horizon. In general this transition function is defined *a priori* using fundamental physical principles and domain knowledge or empirically by applying system identification techniques (Simchowitz et al., 2018) or learning methods from data (Abbeel, Coates & Ng, 2010; Schaal, 1997; Simchowitz et al., 2018). Typically, these methods provide deterministic transition functions that do not incorporate model uncertainty and are invariant over time. As discussed by G. Williams et al. (2017), the closed-loop RHC offers a degree of robustness to model uncertainties, but the compounding error of poor predictions along the control horizon will reduce the stability margins of the system.

Using the methods outlined in Section 2.10, in this chapter we propose a framework to apply the MPPI stochastic control formulation to problems where the parameters of the transition function $f_{\boldsymbol{\xi}}$ are unknown, but belong to a problem dependent *prior distribution*, $p(\boldsymbol{\xi})$. Additionally, we will make use of a LFI method called BayesSim (Ramos, Possas & Fox, 2019) to refine our knowledge of the model parameters as we interact with the environment and gather new observations. The intuition behind this approach is that, by refining our knowledge of the parameters of an otherwise well-defined transition function, we will capitalise not only on the application domain knowledge, but also on the adaptability of inference-free learning methods.

Since $\boldsymbol{\xi}$ represents a plausible range of unknown physical parameters, e.g. mass or friction coefficient, it is straightforward to incorporate domain knowledge to this formulation. Alternatively, an improper uninformative prior may be used when no assumptions are given. Finally, by updating our knowledge of $p(\boldsymbol{\xi} \mid \mathbf{x}^r)$ given observed data, we are more likely to cope with problems such as *covariate shift* (Ganegedara, Ott & Ramos, 2016) and the *reality gap* (Chebotar et al., 2018) discussed in Section 2.11.

3.3 Problem setup

Given a forward model with parameters $\boldsymbol{\xi}$ distributed according to $p(\boldsymbol{\xi})$, trajectories can be obtained from it by first sampling $\boldsymbol{\xi}$ and generating rollouts by propagating the state-action pairs through the transition function. Although the parameters are stochastic, we assume they are invariant throughout the control horizon for a given trajectory. This is a reasonable assumption as the latent parameters are usually stable physical quantities and the update frequency of the control loop is significantly faster than their time constants. For instance, a given system may change its mass over time, but on most cases this change will be negligible during a single pass of the control loop. In this situation, the optimal distribution given in Eq. (2.56) becomes:

$$q^*(V, \boldsymbol{\xi}) = \frac{1}{\eta} \exp\!\left(-\frac{1}{\alpha}\, \mathcal{S}_{\boldsymbol{\xi}}(V)\right) p(V \mid \boldsymbol{\xi})\, p(\boldsymbol{\xi}), \tag{3.3}$$

where we overload the notation to emphasise the dependence of $\mathcal{S}(V, \boldsymbol{\xi})$ on the now stochastic $\boldsymbol{\xi}$. However, as V and $\boldsymbol{\xi}$ are independent[1], we can drop the conditioning in $p(V|\boldsymbol{\xi}) = p(V)$. As a result, our control law can be expressed as:

$$\mathbf{u}_t = \mathbb{E}_{q(\hat{U})}\big[\mathbb{E}_{q(\boldsymbol{\xi})}\big[\omega_{\boldsymbol{\xi}}(V)\mathbf{v}_t\big]\big] = \mathbb{E}_{q(\hat{U},\boldsymbol{\xi})}\big[\omega_{\boldsymbol{\xi}}(V)\mathbf{v}_t\big], \tag{3.4}$$

where $\omega_{\boldsymbol{\xi}}$ shows the dependence on $\boldsymbol{\xi}$ and we applied the law of total expectation to get the resulting equivalence. Eq. (3.4) implies that to compute future control actions, our update rule now has to sample jointly from the distributions of V and $\boldsymbol{\xi}$.

[1]We can safely assume that the parameters of the simulator, $\boldsymbol{\xi}$, do not interfere with the system input, $\mathbf{v}$, and encapsulate all stochasticity into the control mean, $\mathbf{u}$, in the unlikely case such influence exists in practice.

3.4 Propagation of uncertainty over the state-space dynamics

If we sample sufficiently from $p(V, \boldsymbol{\xi})$, we are able to reconstruct the joint distribution $q(V, \boldsymbol{\xi})$ and compute our control updates. In lower dimensional problems, this Monte Carlo approach may be sufficient and constitutes the simplest form of uncertainty propagation. However, we note that the increased dimensionality of the sample space requires the number of samples to grow combinatorially. This combinatorial growth can severely compromise the applicability of our method and therefore we need a more scalable way of propagating model uncertainty over time. As such, we resort to the Unscented Transform (Julier & Uhlmann, 2004) as a more efficient approach to propagate the uncertainty of $\boldsymbol{\xi}$ throughout the state-space.

Once the sigma-points are computed, they are applied recursively to the transition function $f_{\boldsymbol{\xi}}$ to compute the cost $\mathcal{S}(V \mid \boldsymbol{\xi} = \boldsymbol{\chi}_i)$ for $i \in \{1, \ldots, L\}$. In practice, since we assume the simulators parameters are stationary during a rollout, the sigma points on the state space are given by $\boldsymbol{\gamma} = \mathrm{H}[\boldsymbol{\chi}]$. To ensure the trajectory cost of each sigma-point can be summarised using the UT, it is necessary to apply the same action sequence V sampled i.i.d. to all points. Effectively, this means we need to replicate L times each action sequence V during our update step. Finally, the mean trajectory cost is given by

$$\mathcal{S}(V) = \sum_{i=0}^{L} \varpi_i^m \, \mathcal{S}(V \mid \boldsymbol{\xi} = \gamma_i), \tag{3.5}$$

and used in (3.4) for the control law update.

3.5 Updating the parameter prior distribution

At each time step we are computing a new control action $\mathbf{u}_t$, applying it to our environment and collecting new observations $\mathbf{x}_{t+1}^r$. Hence, we can collect the pairs of $[\mathbf{u}_t, \mathbf{x}_t^r]$ in a trajectory dataset $\mathcal{D}$, up to a specified time-length. The trajectory $\mathcal{D}$ may be used as input to estimate the posterior probability of $p(\boldsymbol{\xi} \mid \mathcal{D})$. Many approaches could be used to perform such inference, in this chapter we follow the steps of Ramos, Possas and Fox (2019) and use modern LFI techniques to update the parameters using a black-box simulator to update the posterior estimate once sufficient data has been aggregated in $\mathcal{D}$. In this approach, the posterior distribution over parameters $p(\boldsymbol{\xi} \mid \mathcal{D})$ is approximated by an Mixture Density Network (MDN) parameterised by ϕ denoted by $q_\phi(\boldsymbol{\xi} \mid \mathcal{D})$.

Note that the posterior inference step and the online control-loop need not be executed in tandem. As long as the observations are collected in a trajectory $\mathcal{D}$, the model update can be performed separately and in parallel. In this case, we keep using the prior $p(\boldsymbol{\xi})$ until the approximated posterior $q_\phi(\boldsymbol{\xi} \mid \mathcal{D})$ is available.

Additionally, the unscented transform requires as an input a mean vector $\boldsymbol{\mu}_{\boldsymbol{\xi}}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ for the parameters. Therefore, these have to be retrieved from $q_\phi(\boldsymbol{\xi} \mid \mathcal{D})$, or, alternatively, the highest weighted Gaussian may be selected if it is above a specified threshold. The complete method is presented in Algorithm 2.

3.6 Experimental results

3.6.1 Inverted pendulum swing-up task

In this classic control task the controller goal is to swing and hold a pendulum upright using a torque command applied directly to the joint of a rigid-arm. We used *OpenAI Gym* as the physics simulator (Brockman et al., 2016), and, for each episode, always set the pendulum initial state to the downright position and at rest. The state cost function used was $C_{\text{inst}} = 50\cos(\theta - 1)^2 + \dot{\theta}^2$, and the terminal cost function C_{term} was set to zero. The inverse control temperature α was set at 10 and the control authority $\boldsymbol{\Sigma}$ at 1. We have also defined the number of sampled trajectories $N_{\mathbf{u}} = 500$ and the control horizon $H = 30$. As usual, the minimum control offset term $\tilde{\mathbf{u}}$ was set at zero since this defines the rest state for the controller. The Unscented Transform parameters where chosen as $\alpha = 0.5$, $\kappa = 0$, and $o = 2$. For more details on the experiment parameters, please refer to Appendix A.

The results presented in Fig. 3.1 are the mean cost over time for 50 iterations, for a baseline case using MPPI, DISCO using UT to propagate uncertain rollouts, and DISCO with direct Monte Carlo sampling. It is important to highlight that the baseline case has perfect knowledge of the dynamical system, whereas both instances

Algorithm 2: Double Likelihood-free Inference Stochastic Control

```
1  **Controller Hyperparameters**: $\alpha, \mathbf{\Sigma}, \tilde{\mathbf{u}}, \mathcal{C}_{\text{inst}}, \mathcal{C}_{\text{term}}$;
2  **UT Hyperparameters**: $\nu, \kappa, \alpha, o$;
3  **Given**: $f_{\xi}, p(\xi), U_0, H, N_{\mathbf{u}}, L, \mathcal{D}$;
4  *Update posterior distribution*;
5  $q_{\phi}(\xi \mid \mathcal{D}) \leftarrow \text{BayesSim}(\mathcal{D})$;
6  $p(\xi) \leftarrow q_{\phi}(\xi \mid \mathcal{D})$;
7  **while** *task not complete* **do**
8      $\mathbf{x}_0 \leftarrow \text{GetStateEstimate}()$;
9      **for** $k \leftarrow 0$ **to** $N_{\mathbf{u}} - 1$ **do**
10         Sample $P^k = (\rho_0^k \dots \rho_{H-1}^k)$, for $\rho_t^k \sim \mathcal{N}(0, \mathbf{\Sigma})$;
11         **for** $i \leftarrow 1$ **to** $L$ **do**
12             $\xi_i \leftarrow \xi \sim p(\xi)$ if Monte Carlo, or $\chi_i$ if UT;
13             $\mathbf{x} \leftarrow \mathbf{x}_0$;
14             **for** $t \leftarrow 1$ **to** $H$ **do**
15                 $\mathbf{x} \leftarrow f_{\xi}(\mathbf{x}, \mathbf{v}_t, \xi_i)$;
16                 $\mathcal{S}_i^k \mathrel{+}= \mathcal{C}_{\text{inst}}(\mathbf{x}) + \alpha \mathbf{u}_{t-1}^{\intercal} \mathbf{\Sigma}^{-1} (\rho_{t-1})$;
17             **end**
18             $\mathcal{S}_i^k \mathrel{+}= \mathcal{C}_{\text{term}}(\mathbf{x})$;
19         **end**
20         $\mathcal{S}^k = \sum_{i=1}^{l} \varpi_i^m \mathcal{S}_i^k$;
21     **end**
22     $\tilde{\mathcal{C}} \leftarrow \min \mathcal{S}^k$;
23     $\eta \leftarrow \sum_{k=1}^{N_{\mathbf{u}}} \exp\Big(-\frac{1}{\alpha}\Big(\mathcal{S}^k - \tilde{\mathcal{C}}\Big)\Big)$;
24     **for** $k \leftarrow 1$ **to** $N_{\mathbf{u}}$ **do**
25         $\omega_k \leftarrow \frac{1}{\eta} \exp\Big(-\frac{1}{\alpha}\Big(\mathcal{S}^k - \tilde{\mathcal{C}}\Big)\Big)$;
26     **end**
27     **for** $t \leftarrow 0$ **to** $H - 1$ **do**
28         $\mathbf{u}_t \mathrel{+}= \sum_{k=0}^{N_{\mathbf{u}}-1} \omega(P^k) \rho_t^k$;
29     **end**
30     SendToActuators $(\mathbf{u}_0)$;
31     Append $(\mathcal{D}, [\mathbf{x}_0, \mathbf{u}_0])$;
32     RollControlActions $(\mathbf{u})$;
33 **end**
```

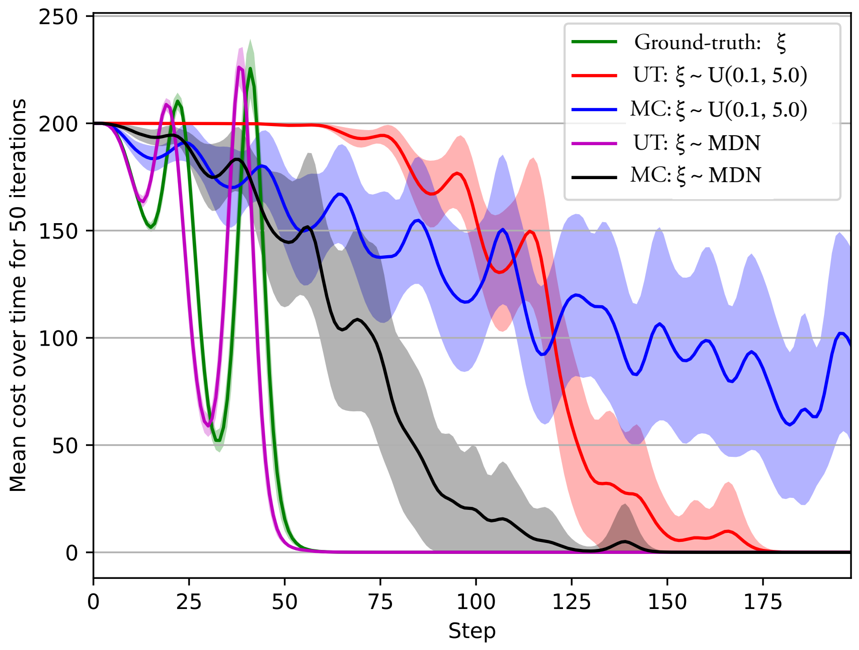


Figure 3.1: **Mean cost over time for the inverted pendulum experiment**. Shaded area represents one standard deviation. Three models where evaluated: a standard MPPI with access to the true system parameters (in green); DISCO using unscented transform with a prior distribution over parameters (in red) and with an updated posterior distribution (in magenta); and DISCO using Monte Carlo sampling with a prior distribution (in blue) and an updated posterior (in black).

using DISCO are using distributions for the length and mass of the arm, as will be discussed shortly. Note that the oscillatory behaviour of the cost function is expected, as the controller has insufficient authority to balance the pendulum without the swinging action to increase the momentum.

All models used the same hyperparameters described above, with the exception of the $N_{\mathbf{u}}$ for the case of Monte Carlo sampling. In order to ensure a fair comparison of the performance of the controller when using UT against Monte Carlo, for the case where Monte Carlo is used we increment the amount of trajectories sampled by a factor equal to the number of sigma points L used by the UT. In other words, for the Monte Carlo controller we have effectively $N_{\mathbf{u}} = 2500$ trajectories.

The unknown parameters in this example were the length of the arm and the mass of the pendulum. As a prior, we assumed an uniform distribution between 0.1 and 5 for both parameters. The posterior distribution was given by a mixture of Gaussians with 5 components, trained using a reference control policy. Note that in our simulations, both models shared the same posterior distribution estimate. Once trained and conditioned on the observed data, the resulting mixture had a mean estimate for the length of $0.89\,\mathrm{m}$ and $0.9\,\mathrm{kg}$ for mass. The covariance matrix was diagonal, and the variance was $0.01\,\mathrm{m}^2$ for the length estimate and $0.03\,\mathrm{kg}^2$ for mass. One of the components of the mixture was dominant with a weight of 0.979 and was used as reference for the UT.

DISCO with UT outperforms Monte Carlo sampling both with an uninformative prior and inferred posterior. Noticeable also, the performance of UT with the posterior distribution is better than the baseline model. This is explainable by the fact that the parameter randomisation introduced by the sigma-points provides more information in the trajectory evaluation. This way, trajectories that are borderline to a higher cost state captured by one of the sigma points get penalised. Effectively, UT works like an automatic calibration of the control temperature, when the prior is broad, many trajectories are considered in the control update average. Conversely, when the posterior gets refined, the controller is more confident to select fewer trajectories.

3.6.2 Skid-steer robot

This section presents experimental results with a physical robot equipped with a skid-steering drive mechanism (Fig. 3.2). We modelled the kinematics of the robot based on a modified unicycle model, which accounts for skidding via an additional parameter (Kozłowski & Pazderski, 2004). The parameters to be estimated via BayesSim are the robot's wheel radius, r_w, axial distance, i.e. the distance between the wheels, given by a_w, and the displacement of the robot's Inertial Centre of Rotation (ICR), x_ICR, from the robot's centre. A non-zero value on the latter affects turning by sliding the robot sideways. To estimate the parameters, the robot was driven manually around a circle and had its trajectory data recorded. From the trajectory data we computed cross-correlation summary statistics as $(\overline{x}, \overline{y}, \overline{\Delta x}, \overline{\Delta y})$, which capture the centre of trajectory and the average linear velocity. In simulation, the wheel speed commands sent to the robot were repeated $N = 1000$ for different parameter settings sampled from

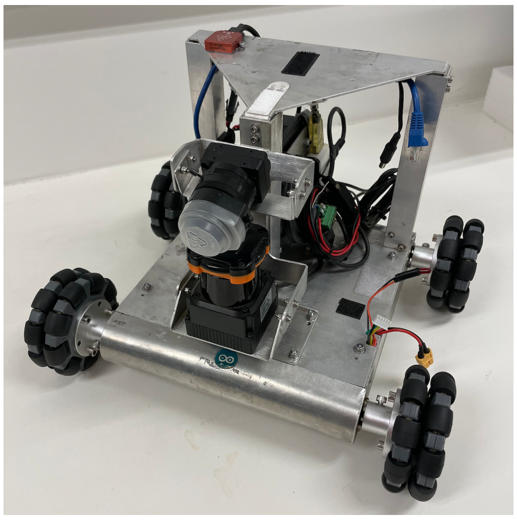


Figure 3.2: **The *Wombot***, a custom built skid-steer robot used in many experiments. The robot is speed-controlled with separate set-points for left and right wheels speed.

a uniform prior, $x_{\mathrm{ICR}} \sim \mathcal{U}(0, 0.5)$, $r_{\mathrm{w}} \sim \mathcal{U}(0, 0.5)$, $a_{\mathrm{w}} \sim \mathcal{U}(0.1, 0.5)$.

Figure 3.3 presents the resulting marginal estimates from BayesSim for each parameter of the robot's kinematic model. For comparisons, physical measurements indicate a r_{w} of around $0.06\,\mathrm{m}$ and a_{w} of around $0.31\,\mathrm{m}$. Measuring x_{ICR}, however, involves a laborious process, which would require different weight measurements or many trajectories from the physical hardware (Yi et al., 2009). As we are only applying a relatively simple kinematic model of the robot to explain the real trajectories, the effects of the dynamics and ground-wheel interactions are not accounted for. As a result, BayesSim tries to compensate for the miss-specifications in some parameters estimation, such as the axial distance. This explains the larger variation in a_{w}, and consequently x_{ICR}.

The control task was defined as following a circular path at a constant tangential speed. Costs were set to make the robot follow a circle of $0.75\,\mathrm{m}$ radius with $\mathcal{C}(\mathbf{x}_t) = \sqrt{d_t^2 + (\dot{\mathbf{x}}_t - \dot{\mathbf{x}}_0)^2}$, where d_t represents the robot's distance to the edge of the circle and $\dot{\mathbf{x}}_0 = 0.2\,\mathrm{m\,s^{-1}}$ is a reference linear speed. We performed experiments sampling from the uniform prior over the parameters $p(\boldsymbol{\xi})$, sampling from the posterior $q_{\phi}(\boldsymbol{\xi} \mid \mathcal{D})$, and using only a point estimate set to $x_{\mathrm{ICR}} = 0.12\,\mathrm{m}$, $r_{\mathrm{w}} = 0.06\,\mathrm{m}$ and $a_{\mathrm{w}} = 0.47\,\mathrm{m}$, which was adjusted offline to reduce simulation error. For clarity, instead of the noisy raw costs, we present the mean instant cost, i.e.

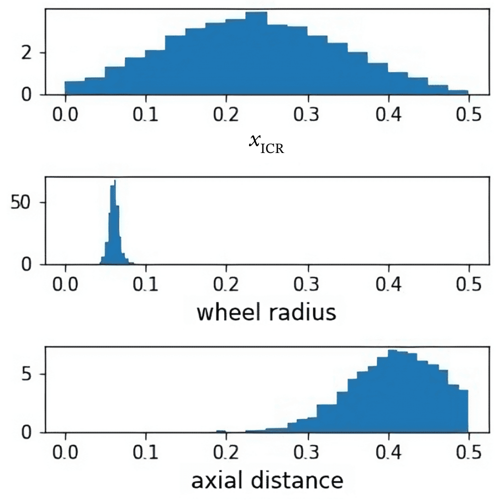


Figure 3.3: **Estimated parameters found by BayesSim** for the Wombot skid-steer robot after analysing the manually collected dataset. Parameters such as x_{ICR} and a_{w} can be confounding factors and therefore the difficulty in measuring x_{ICR} impacts the uncertainty of a_{w}. In order to measure x_{ICR} with higher precision, a higher-fidelity physics model needed to be used.

$\overline{C_{\text{inst}}}_t = \frac{1}{t}\sum_{i=1}^{t} C_{\text{inst}}(\mathbf{x}_i)$ and the executed trajectories in Fig. 3.4. For the complete experiment parameters please refer to Appendix A.

We see that considering parameter uncertainty via DISCO provides significant performance improvements over the baseline MPPI algorithm running with a point estimate. Although, in term of costs, both the prior and posterior estimates offer similar performance, we see the advantages of using the parameter posterior estimates in the trajectories plot, where the overshooting happening on some portions of the path is drastically reduced. The latter can be explained by the prior allowing kinematic parameters candidates that are too far from the true values. Additionally, refining the posterior distribution allows the system to adapt to new configurations or drift in the model parameters. Lastly, a noisier speed control explains the gap between the baseline MPPI and the DISCO methods, despite the similar performances in terms of path tracking.

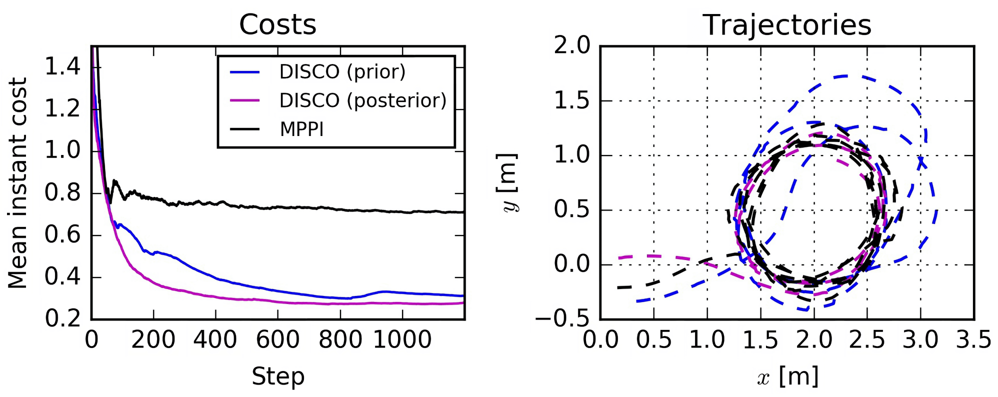


Figure 3.4: **Experimental results of applying DISCO to the physical robot**. Note how the controller is able to cope relatively well with a broad prior distribution over model parameters. Moreover, once the posterior distribution is refined, the quality of the trajectories improves significantly, with reduced overshooting and quality comparable to our baseline case with fine-tuned parameters.

3.7 Summary

In this chapter we presented a first step towards incorporating model uncertainty and sophisticated Bayesian inference methods to stochastic model based control. We showed how uncertainty over parameters may be formally incorporated into an SN-MPC controller and evaluated methods of propagating the uncertainty into trajectory rollouts. This extension to Model Predictive Path Integral provides the building blocks of an adaptive controller framework, more resilient to issues arising from reality gap and covariate shift. As shown in the robotic experiments, incorporating uncertainty may lead to a more accurate assessment of the environment and increase the performance.

The unscented transform proved an efficient way to propagate uncertainty, reducing the burden on trajectory samples. Since there is no assumption on the smoothness of the cost function and no differentiability requirement, we are able to impose hard-penalties on the violation of soft constraint. It follows that the combined effect is similar to a chance constraint where the resulting trajectories from sigma points that break some of the desired constraints are heavily penalised. It is worth noticing, that this deterministic method of estimating the moments of the parameter distribution allow the task of sampling actions to be parallelised asynchronously and aggregated when computing the final cost.

However, there is a trade-off between the efficiency gained in sampling and accuracy when using moment matching. More precisely, we are approximating the posterior distribution of trajectories over control policies and model parameters with an uni-modal Gaussian distribution whose first and second order moments match those of the full posterior distribution. Intuitively, we expect that the Unscented Transform should suffer whenever the posterior distribution is multi-modal, as is the case in many practical instances. In Chapter 4 we will explore these limitations and discuss alternatives.

Finally, we showed how LFI is a powerful tool to refine the estimation of the posterior distribution. As the inference is based on the same transition function f_{ξ} used by the controller, it may compensate overly simplified models of the environment. This is often the case, since the internal model used by MPC needs to be efficient in order to generate a large number of real-time rollouts. Therefore, in the following chapters we want to explore pathways to efficiently retrain this estimate online so practical experiments with time-variant parameters may be conducted. This is a crucial step towards generalisation of control policies for autonomous robots operating under varying environments and configurations. Crucially, by combining parameter estimation and gradient-free control methods, DISCO may also be used with black-box simulators, such as data-driven function approximators, as long as we are able to sample efficiently from them.

Choices are the hinges of destiny.

Edwin Markham

CHAPTER 4

Online inference of actions and model parameters

In Chapter 3 we have seen how real robotics applications are invariably subjected to uncertainty arising from either unknown model parameters or stochastic environments. We have also discussed how MPC is one of the most used approaches to design controllers robust to such uncertainty. In its essence, MPC relies on different optimisation strategies to find a sequence of actions over a given control horizon that minimises an optimality criteria defined by a cost function.

Furthermore, despite its success in practical applications, most MPC approaches do not take uncertainty into account. Traditional dynamic-programming approaches to MPC, such as Iterative Linear Quadratic Regulator (iLQR) (Tassa, Erez & Todorov, 2012) and DDP (Tassa, Mansard & Todorov, 2014), rely on a differentiable and deterministic cost function and dynamics model. Stochastic Optimal Control variants, such as Iterative Linear Quadratic Gaussian (iLQG), by Todorov and Weiwei Li (2005) and Probabilistic Differential Dynamic Programming (PDDP), by Pan and Theodorou (2014), can accommodate stochastic dynamics, but only under simplifying assumptions such as additive Gaussian noise. These approaches are generally less effective in addressing complex distributions over actions, and it is unclear how these methods should incorporate model uncertainty, if any.

In contrast, sampling-based control schemes have gained increasing popularity for their general robustness to model uncertainty, ease of implementation, and ability to contend with sparse cost functions (G. Williams, Goldfain et al., 2018). In sampling based Stochastic Optimal Control (SOC), the optimisation problem is replaced by a sampling-based algorithm that tries to approximate an optimal distribution over actions (G. Williams, Aldrich & Theodorou, 2017). We have seen in Chapter 3 how approaches like DISCO have shown promising results in several applications and addresses some of the disadvantages of traditional MPC approaches, however it too has drawbacks. Most notably, the posterior distribution is only approximated through moment-matching, resulting in a uni-modal estimate that may inadequately capture the complexity of the true posterior.

In recent work Lambert et al. (2020) and Okada and Taniguchi (2020) reformulated the MPC problem as a Bayesian inference task whose goal is to estimate the posterior distribution over control parameters given the state and observed costs. To make such problem tractable in real-world applications, variational inference has been used to approximate the optimal distribution. These approaches are better suited at handling the multi-modality of the distribution over actions, but do not attempt to dynamically adapt to changes in the environment.

Conversely, previous work has demonstrated that incorporating uncertainty in the evaluation of SOC estimates can improve performance (Barcelos et al., 2020), particularly when this uncertainty is periodically re-estimated (Possas et al., 2020). Although this method is more robust to model mismatch and help address the sim-to-real gap, the strategy relies on applying moment-matching techniques to propagate the uncertainty through the dynamical system which is approximated by a Gaussian distribution. This diminishes the effectiveness of the method under settings where multi-modality is prominent.

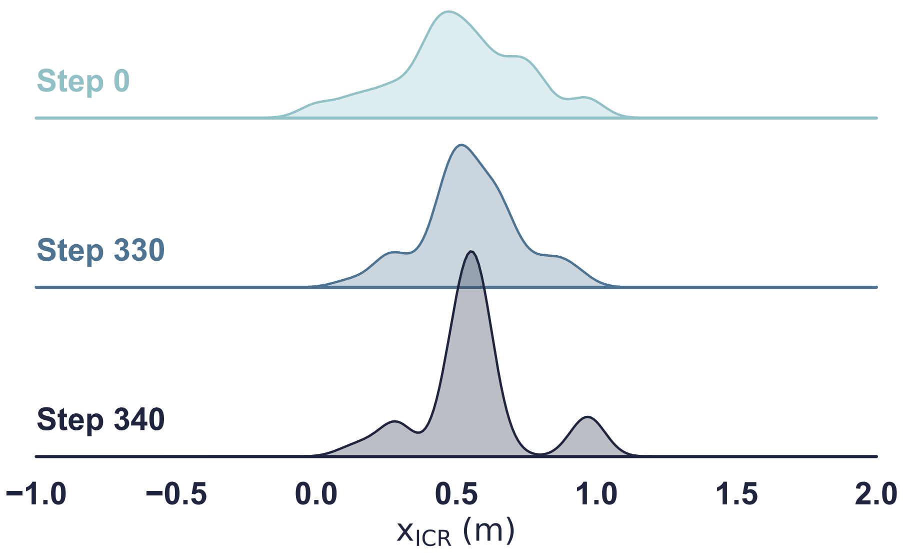
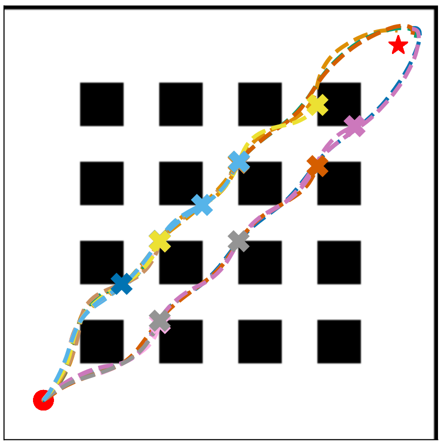
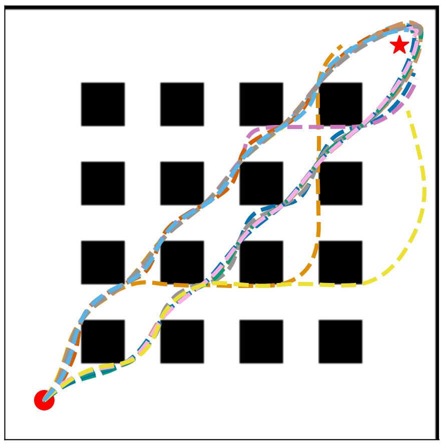


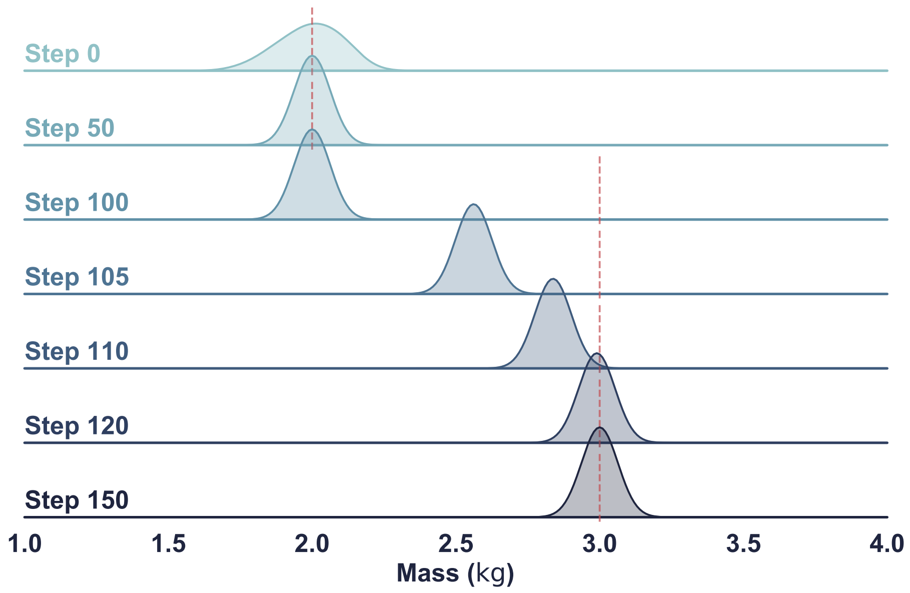
Figure 4.1: **Online parameter estimation for autonomous ground vehicles**. DuSt-MPC is able to reason and update distributions over system parameters in real-time. The ridge plot shows the distribution over x_{ICR} for the Wombot (see Fig. 3.2) at different time steps. The load mass on the robot is suddenly increased during execution and the parameter distribution estimate quickly changes to include a second mode that better explains the new dynamics. Our particle-based control scheme can leverage such multi-modal distribution and hence adapt to dynamically changing environments.

In this chapter, we aim to leverage recent developments in variational inference with MPC for decision-making under complex multi-modal uncertainty over actions while simultaneously estimating the uncertainty over model parameters. We propose a Stein variational stochastic gradient solution that models the posterior distribution over actions and model parameters as a collection of particles, representing an implicit variational distribution. These particles are updated sequentially, online, in parallel, and can capture complex multi-modal distributions. We call this method Dual Stein Variational Inference MPC, or Dual Stein Variational Model Predictive Control (DuSt-MPC) for conciseness[1].

[1]Note, that *dual* is used in the sense of a doubly stochastic problem, i.e. inferring a distribution over policies and system dynamics, and not as a duality problem in optimisation.


(a) SVMPC trajectories (baseline)


(b) DuSt-MPC trajectories



(c) Ridge plot of mass distribution (in kg)

Figure 4.2: **Point-mass navigation task**. The plots shows trajectories from the start position (red dot) towards the goal (red star). **Top-Left**: Trajectories executed by SVMPC. Note that, as the mass of the robot changes, the model mismatch causes many of the episodes to crash (**x** markers). **Top-Right**: Trajectories executed by DuSt-MPC. Depending on the state of the system when the mass change occurs, a few trajectories deviate from the centre path to avoid collisions. A few trajectories are truncated due to the fixed episode length. **Bottom**: Ridge plot of the distribution over mass along several steps of the simulation. The vertical dashed line denotes the true mass. Mass is initially set at 2 kg, and changed to 3 kg at step 100.

Specifically, the main contributions of this chapter are:

- We propose a principled Bayesian solution of introducing uncertainty over the model parameters in Stein variational MPC and empirically demonstrate how this can be leveraged to improve the control policy robustness;
- We introduce a novel method to extend the inference problem and simultaneously optimise the control policy while refining our knowledge of the environment as new observations are gathered. Crucially, by leveraging recent advancements in sequential Monte Carlo with kernel embedding, we perform online, sequential updates to the distribution over model parameters which scales to large datasets;
- By capturing the uncertainty on true dynamic systems in distributions over a parametric model, we are able to incorporate domain knowledge and physics principles while still allowing for a highly representative model. This simplifies the inference problem and drastically reduces the number of interactions with the environment to characterise the model.

We implement the algorithm on a real Automated Guided Vehicle (AGV), (see Fig. 3.2), illustrating the applicability of the method in real time. Experiments show how the control and parameter inference are leveraged to adapt the behaviour of the robot under varying conditions, such as changes in mass. We also present simulation results on an inverted pendulum and an 2-D obstacle grid, see Fig. 4.2, demonstrating an effective adaptation to dynamic changes in model parameters.

This chapter is organised as follows. In Section 4.1 we review related work, contrasting the proposed method to the existing literature. The main method is presented from Sections 4.2 to 4.5 and relies on the foundational knowledge presented in Sections 2.6 and 2.9. In Section 4.6 we present a number of real and simulated experiments, followed by relevant conclusions in Section 4.7.

4.1 Related work

Sampling-based approaches for stochastic MPC have shown to be suitable for a range of control problems in robotics (Wagener et al., 2019; G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018). At each iteration, these methods perform an approximate evaluation by rolling-out a stochastic policy with modelled system dynamics over a finite-length horizon. The optimisation step proceeds to update the policy parameters in the direction that minimises the expected cost and a statistical distance to a reference policy or prior (Wagener et al., 2019). This can equivalently be interpreted as a statistical inference procedure, where policy parameters are updated in order to match an optimal posterior distribution (Agarwal et al., 2013; Lambert et al., 2020; G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018). This connection has motivated the use of common approximate inference procedures for SOC. MPPI (G. Williams, Drews, Goldfain, Rehg & Theodorou, 2018; G. Williams, Aldrich & Theodorou, 2017) and the Cross Entropy Method (CEM) (Botev et al., 2013), for instance, use importance sampling to match a Gaussian proposal distribution to moments of the posterior. VI approaches have also been examined for addressing control problems exhibiting highly non-Gaussian or multi-modal posterior distributions. This family of Bayesian inference methods extends the modelling capacity of control distribution to minimise a Kullback-Leibler divergence with the target distribution. Traditional VI methods such as Expectation-Maximisation have been examined for control problems (Okada & Taniguchi, 2020; Watson, Abdulsamad & Peters, 2020), where the model class of the probability distribution is assumed to be restricted to a parametric family (typically Gaussian Mixture Models). More recently, Lambert et al. proposed to adapt SVGD (Liu & Wang, 2016) for use in model predictive control. Here, a distribution of control sequences is represented as a collection of particles. The resulting framework, Stein Variational Model Predictive Control (SVMPC), adapts the particle distribution in an online fashion. The non-parametric representation makes the approach particularly suitable to systems exhibiting multi-modal posteriors. In the present work, we build on the approach in (Lambert et al., 2020) to develop an MPC framework which leverages particle-based variational inference to optimise controls *and* explicitly address model uncertainty. Our method *simultaneously* adapts a distribution over dynamics parameters, which we demonstrate improves robustness and performance on a real system.

Model predictive control is a reactive control scheme, and can accommodate modelling errors to a limited degree. However, its performance is largely affected by the accuracy of long-range predictions. Modelling errors can compound over the planning horizon, affecting the expected outcome of a given control action. This can be mitigated by accounting for model uncertainty, leading to better estimates of expected cost. This has been demonstrated to improve performance in stochastic optimal control methods and model-based reinforcement learning (Deisenroth & Rasmussen, 2011; Pan & Theodorou, 2014). Integrating uncertainty has typically been achieved by learning probabilistic dynamics models from collected state-transition data in an episodic setting, where the model is updated in between trajectory-length system executions (Chua et al., 2018; Okada & Taniguchi, 2020; Ramos, Possas & Fox, 2019; Wabersich & Zeilinger, 2020). A variety of modelling representations have been explored, including Gaussian processes (Deisenroth & Rasmussen, 2011), neural network ensembles (Chua et al., 2018), Bayesian regression, and meta-learning (Harrison, Sharma & Pavone, 2020). Additionally, Ramos, Possas and Fox (2019) used blackbox simulators to estimate the posterior distributions of physical parameters given real-world observations.

Recent efforts have examined the online setting, where a learned probabilistic model is updated based on observations made *during execution* (Abraham et al., 2020; Fan, Agha & Theodorou, 2020; Fisac et al., 2019). The benefits of this paradigm are clear: incorporating new observations and adapting the dynamics *in situ* will allow for better predictions, improved control, and recovery from sudden changes to the environment. However, real-time requirements dictate that model adaptation must be done quickly and efficiently, and accommodate the operational timescale of the controller. This typically comes at the cost of modelling accuracy, and limits the application of computationally-burdensome representations, such as neural networks and vanilla GPs. Previous work has included the use of sparse-spectrum GPs and efficient factorization to incrementally update the dynamics model (Pan et al., 2017). In (Harrison, Sharma & Pavone, 2020), the authors use a meta-learning approach to train a network model offline, which is adapted to new observations using Bayesian linear regression operating on the last layer. However, these approaches are restricted to Gaussian predictive distributions, and may lack sufficient modelling power for predicting complex, multi-modal distributions.

Perhaps most closely related to our modeling approach is the work by Abraham et al. (2020). The authors propose to track a distribution over simulation parameters using a sequential Monte Carlo method akin to a particle filter. The set of possible environments resulting from the parameter distribution is used by an MPPI controller to generate control samples. Each simulated trajectory rollout is then weighted according to the weight of the corresponding environment parameter. Although such an approach can model multi-modal posterior distributions, we should expect similar drawbacks to particle filters, which require clever re-sampling schemes to avoid mode collapse and particle depletion. Our method also leverages a particle-based representation of parameter distributions, but performs deterministic updates based on new information and is more sample efficient than Monte Carlo sampling techniques.

In the next sections, we present our MPC approach for joint inference over control and model parameters. We begin by formulating optimal control as an inference problem and address how to optimise policies in Section 4.4. Later, in Section 4.5, we extend the inference to also include the system dynamics. A complete overview of the method is shown in algorithm Algorithm 3.

4.2 MPC as Bayesian inference

MPC can be framed as a Bayesian inference problem where we estimate the posterior distribution of policies, parameterised by θ_t, given an optimality criterion. Note how we index the policy parameters by t to indicate that they may vary over time. Now let $\mathcal{O} : \mathcal{P}_{\mathcal{X}} \to \{0, 1\}$ be an *optimality* indicator for a trajectory $\tau \in \mathcal{P}_{\mathcal{X}}$ such that $\mathcal{O}[\tau] = 1$ indicates that the trajectory is optimal. Now we can apply Bayes' Rule and frame our control problem as estimating:

$$p(\theta_t \mid \mathcal{O}) \propto p(\mathcal{O} \mid \theta_t)\, p(\theta_t) = p(\mathcal{O}, \theta_t). \tag{4.1}$$

In essence, optimality is a subjective term which might be difficult to define. However, we may agree that in the MPC formulation the optimality of a trajectory is related to the task-dependent cost functional defined in Section 2.8. A reasonable to way to quantify $\mathcal{O}[\tau]$ is to model it as a Bernoulli random variable conditioned on the trajectory τ, allowing us to define the *likelihood* $p(\mathcal{O} \mid \theta_t)$ as:

$$p(\mathcal{O}[\tau] = 1 \mid \tau) := \exp(-\alpha\, \mathcal{C}[\tau]), \tag{4.2}$$

where $\alpha \in \mathbb{R}^+$ is a constant and $\mathcal{C}[\tau]$ is the cost functional measuring optimality. Since $\mathcal{O}$ is an indicator functional over trajectories, it also defines a set of optimal trajectories:

$$\mathcal{P}_{\mathcal{X}}^{*} := \{\tau \in \mathcal{P}_{\mathcal{X}} \mid \mathcal{O}[\tau] = 1\} \subset \mathcal{P}_{\mathcal{X}}. \tag{4.3}$$

We can measure the probability of generating trajectories in the optimal set $\mathcal{P}_{\mathcal{X}}^{*}$ with a policy π_{θ_t} as:

$$\begin{aligned} \mathrm{P}(\mathcal{P}_{\mathcal{X}}^{*} \mid \theta_t) &= \int_{\mathcal{P}_{\mathcal{X}}} p(\mathcal{O}[\tau] = 1, \tau \mid \theta_t)\, \mathrm{d}\tau \\ &= \int_{\mathcal{P}_{\mathcal{X}}} p(\mathcal{O}[\tau] = 1 \mid \tau)\, p(\tau \mid \theta_t)\, \mathrm{d}\tau. \end{aligned} \tag{4.4}$$

Note that the above $\mathrm{P}(\mathcal{P}_{\mathcal{X}}^{*} \mid \theta_t)$ denotes the probability *measure*, not the probability density, of the set $\mathcal{P}_{\mathcal{X}}^{*}$. In practice, we cannot integrate over $\mathcal{P}_{\mathcal{X}}$, but we can generate trajectories via independent rollouts $\tau_i \sim p(\tau \mid \theta_t)$, for $i \in \{1, \dots, N\}$. We then define and approximate the parameters likelihood as

$$\begin{aligned} p(\mathcal{O} \mid \theta_t) &:= \mathrm{P}(\mathcal{P}_{\mathcal{X}}^{*} \mid \theta_t) \\ &= \mathbb{E}_{\tau \sim p(\tau \mid \theta_t)}\big[p(\mathcal{O}[\tau] = 1 \mid \tau)\big] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \exp(-\alpha\, C[\tau_i]), \end{aligned} \tag{4.5}$$

where we overload $p(\mathcal{O}[\tau] = 1 \mid \theta_t)$ to simplify the notation. The variable α is known as the inverse temperature parameter and controls the amount of exploration of the policy. A lower α results in smaller separation between each cost functional and encourages more trajectories to contribute in the gradient step. Conversely, higher values of α increase the cost functional separation of each rollout and promotes a more exploitative policy. Now, assuming a prior $p(\theta_t)$ for θ_t, the posterior over θ_t is given by:

$$\begin{aligned} p(\theta_t \mid \mathcal{O}) &\propto p(\mathcal{O} \mid \theta_t)\, p(\theta_t) \\ &\propto \int_{\mathcal{P}_{\mathcal{X}}} p(\mathcal{O} \mid \tau)\, p(\tau \mid \theta_t)\, p(\theta_t)\, \mathrm{d}\tau. \end{aligned} \tag{4.6}$$

This posterior corresponds to the probability density of a given parameter setting θ_t conditioned on the hypothesis that (implicitly observed) trajectories generated by θ_t are *optimal*. Alternatively, one may say that $p(\theta_t \mid \mathcal{O})$ tells us the probability of θ_t

being the generator of the optimal set $\mathcal{P}^*_{\mathcal{X}}$. Lastly, note that the trajectories conditional distribution $p(\tau \mid \theta_t)$ factorises as:

$$p(\tau \mid \theta_t) = \prod_{h=0}^{H-1} p_{\xi}[\mathbf{x}_{t+h+1} \mid \mathbf{x}_{t+h}, \mathbf{u}_{t+h}]\, \pi_{\theta_t}(\mathbf{x}_{t+h}), \tag{4.7}$$

where H is the control horizon, p_ξ is the probability density of the parameterised transition function and π_{θ_t} is the control policy from which $\mathbf{u}_{t+h}$ is sampled. This factorisation will be used in the next section, in which we explore how to perform joint inference of control policy and system dynamics.

4.3 Joint inference of policy and dynamics

In this section, we generalise the framework presented in Lambert et al. (2020) to simultaneously refine our knowledge of the dynamical system, parameterised by ξ, while estimating optimal policy parameters θ_t. Before we proceed, however, it is important to notice that the optimality measure defined in Section 4.2 stems from *simulated* rollouts sampled according to Eq. (4.5). Hence, these trajectories are not actual observations of the agent's environment and are not suitable for inferring the parameters of the system dynamics. In other words, to perform inference over the parameters ξ we need to collect *real observations* from the environment.

Moreover, we argue that the two inference problems can be naturally factorised. From the perspective of the plant, i.e. the physical system, the dynamics inference depends solely on previously observed data, which despite including past control actions, is independent of the current policy. On the other hand, the policy inference assumes that the distribution over the system dynamics at a given time is given and invariant and relies exclusively on simulated future rollouts. With that in mind, the problem statement in Eq. (4.1) can be rewritten as:

$$p(\theta_t, \xi \mid \mathcal{O}, \mathcal{D}_{1:t}) = p(\theta_t \mid \mathcal{O}, \xi)\, p(\xi \mid \mathcal{D}_{1:t}), \tag{4.8}$$

where $\mathcal{D}_{1:t} := \{(\mathbf{x}^r_t, \mathbf{u}^r_{t-1}, \mathbf{x}^r_{t-1})\}_{t=1}^N$ represents the dataset of collected environment observations. Note that the distribution over the parameters of the plant, the dynamical system, is independent from the policy optimality, and therefore the conditioning on $\mathcal{O}$ has been dropped. Similarly, once the distribution over ξ is defined, the policy parameters $\boldsymbol{\theta}$ are independent from the previous environment observations $\mathcal{D}_{1:t}$, and again we omit the conditioning.

If one were to solve a joint inference problem by defining a new random variable $\Theta = \{\boldsymbol{\theta}, \boldsymbol{\xi}\}$, and following the steps outlined by Lambert et al. (2020), careful consideration should be taken to prevent the partial derivative of $\boldsymbol{\xi}$ from including terms dependent on the policy optimality (see Appendix B.4 for further discussion). Furthermore, by factorising the problem we can perform each inference step separately and adjust the computational effort and hyper-parameters based on the idiosyncrasies of the problem at hand.

4.4 Policy inference for Bayesian MPC

Having formulated the inference problem as in Eq. (4.8), we can proceed by solving each of the factors separately. We shall start with optimising $\pi_{\boldsymbol{\theta}_t}$ according to $p(\boldsymbol{\theta}_t \mid \mathcal{O}, \boldsymbol{\xi})$. It is evident that we need to consider the dynamics parameters when optimising the policy, but at this stage we may simply assume the inference over $\boldsymbol{\xi}$ has been solved and leverage our knowledge of $p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$. More concretely, let us rewrite the factor on the RHS of Eq. (4.8) by marginalising over $\boldsymbol{\xi}$ so that:

$$p(\boldsymbol{\theta}_t \mid \mathcal{O}, \mathcal{D}_{1:t}) \propto \int_{\Xi} \ell_\pi(\boldsymbol{\theta}_t)\, p(\boldsymbol{\theta}_t)\, p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})\, \mathrm{d}\boldsymbol{\xi}, \tag{4.9}$$

where, as in Eq. (4.5), the likelihood $\ell_\pi(\boldsymbol{\theta}_t)$ is defined as:

$$\begin{aligned} \ell_\pi(\boldsymbol{\theta}_t) &= p(\mathcal{O} \mid \boldsymbol{\theta}_t, \boldsymbol{\xi}) := \mathrm{P}(\mathcal{P}^*_{\mathcal{X}} \mid \boldsymbol{\theta}_t, \boldsymbol{\xi}) \\ &= \int_{\mathcal{P}_{\mathcal{X}}} p(\mathcal{O} \mid \boldsymbol{\tau})\, p(\boldsymbol{\tau} \mid \boldsymbol{\theta}_t, \boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\tau} \\ &\approx \frac{1}{N_\tau} \sum_{i=1}^{N_\tau} \exp(-\alpha\, C[\boldsymbol{\tau}_i]), \end{aligned} \tag{4.10}$$

with $\boldsymbol{\tau}_i \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}_t, \boldsymbol{\xi})$, $i \in \{1, 2, \dots, N_\tau\}$, and $p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$ defines the inference problem of updating the posterior distribution of the dynamics parameters given all observations gathered from the environment, which we shall discuss in the next section. Careful consideration of Eq. (4.10) tells us that unlike the case of a deterministic transition function or even maximum likelihood point estimation of $\boldsymbol{\xi}$, the optimality of a given trajectory now depends on its *expected* cost over the distribution $p(\boldsymbol{\xi})$.

Hence, given a prior $p(\theta_t)$ and $p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$, we can generate samples from the likelihood in Eq. (4.10) and use an stochastic gradient method as in Section 2.6 to infer the posterior distribution over θ_t. Following the steps of Lambert et al. (2020), we approximate the prior $p(\theta_t)$ by a set of particles $q(\theta_t) = \{\theta_t\}_{i=1}^{N_\pi}$ and take sequential SVGD updates:

$$\theta_t^i \leftarrow \theta_t^i + \epsilon\, \boldsymbol{\phi}^*(\theta_t^i), \tag{4.11}$$

to derive the posterior distribution. Where, again, $\boldsymbol{\phi}^*$ is computed as in Eq. (2.40) for each intermediate step and ϵ is a predetermined step size. One ingredient missing to compute the score function is the gradient of the log-posterior of $p(\theta_t \mid \mathcal{O}, \boldsymbol{\xi})$, which can be factorised into:

$$\nabla_{\theta_t^i} \log p(\theta_t^i \mid \mathcal{O}, \boldsymbol{\xi}) = \nabla_{\theta_t^i} \log \ell(\theta_t^i) + \nabla_{\theta_t^i} \log q(\theta_t^i). \tag{4.12}$$

In practice we typically assume that the $\mathcal{C}[\cdot]$ functional used as surrogate for optimality is not differentiable w.r.t. θ_t, but the gradient of the log-likelihood can be usually approximated via Monte Carlo sampling.

Most notably, however, is the fact that unlike the original formulation in SVGD, the policy distribution we are trying to infer is time-varying and depends on the actual state of the system. The measure $\mathrm{P}(\mathcal{P}_{\mathcal{X}}^* \mid \theta_t, \mathbf{x}_t)$ depends on the current state, as that is the initial condition for all trajectories evaluated in the cost functional $\mathcal{C}[\tau]$.

Theoretically, one could choose an uninformative prior with broad support over the $\mathcal{U}$ at each time-step and employ standard SVGD to compute the posterior policy. However, due to the sequential nature of the MPC algorithm, it is likely that the prior policy $q(\theta_{t-1})$ is close to a region of low cost and hence is a good candidate to bootstrap the posterior inference of the subsequent step. This procedure is akin to the prediction step commonly found in Sequential Bayesian Estimation (Doucet, 2001). More concretely,

$$q(\theta_t) = \int p(\theta_t \mid \theta_{t-1})\, q(\theta_{t-1})\, \mathrm{d}\theta_{t-1}, \tag{4.13}$$

where $p(\theta_t \mid \theta_{t-1})$ can be an arbitrary transitional probability distribution. More commonly, however, is to use a probabilistic version of the shift operator as defined in (Wagener et al., 2019). For a brief discussion on action selection, please refer to

Appendix B.2. For more details on this section in general the reader is encouraged to refer to (Lambert et al., 2020, Sec. 5.3).

4.5 Real-time dynamics inference

We now focus on the problem of updating the posterior over the simulator parameters. Note that, due to the independence of each inference problem, the frequency in which we update $p(\boldsymbol{\xi})$ can be different from the policy update. In fact, as discussed in Section 4.1, many previous works rely on this to refine the parametric transition function in an *episodic* setting (Okada & Taniguchi, 2020; Possas et al., 2020; Ramos, Possas & Fox, 2019). Then, recursively, the new parameter distribution can be used during a new episode of data collection.

In contrast, we are interested in the case where $p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$ can be updated in *real-time* adjusting to changes in the environment. For that end, we need a more efficient way of updating our posterior distribution. The inference problem at a given time-step can then be written as:

$$p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t}) \propto p(\mathcal{D}_t \mid \boldsymbol{\xi}, \mathcal{D}_{1:t-1})\, p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t-1}). \tag{4.14}$$

Note that in this formulation, $\boldsymbol{\xi}$ is considered time-invariant. This based on the implicit assumption that the frequency in which we gather new observations is significantly larger than the covariate shift to which $p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$ is subject to as we traverse the environment. In Appendix B.3 we discuss the implications of changes in the latent parameter over time.

In general, we do not have access to direct measurements of $\boldsymbol{\xi}$, only to the system state. Therefore, in order to perform inference over the dynamics parameters, we rely on a generative model, i.e. the simulator $\hat{f}_{\boldsymbol{\xi}}$, to generate samples in the state space $\mathcal{X}$ for different values of $\boldsymbol{\xi}$. However, unlike in the policy inference step, for the dynamics parameter estimation we are not computing deterministic simulated rollouts, but rather trying to find the explanatory parameter for each observed transition in the environment. Namely, we have:

$$\mathbf{x}_t^r = f(\mathbf{x}_{t-1}^r, \mathbf{u}_{t-1}) = \hat{f}_{\boldsymbol{\xi}}(\mathbf{x}_{t-1}^r, \mathbf{u}_{t-1}) + \eta_t, \tag{4.15}$$

where $\mathbf{x}_t^r$ denotes the true system state and η_t is a time-dependent random variable closely related to the *reality gap* in the sim-to-real literature (Valassakis, Ding & Johns, 2020) and incorporates all the complexities of the real system not captured in simulation, such as model mismatch, unmodelled dynamics, etc. As a result, the distribution of η_t is unknown, correlated over time and hard to estimate.

In practice, for the feasibility of the inference problem, we make the standard assumption that the noise is distributed according to a time-invariant normal distribution $\eta_t \sim \mathcal{N}(0, \mathbf{\Sigma}_{\text{obs}})$, with an empirically chosen covariance matrix. More concretely, this allows us to define the likelihood term in Eq. (4.14) as:

$$\begin{aligned} \ell(\boldsymbol{\xi} \mid \mathcal{D}_{1:t}) &\coloneqq p(\mathcal{D}_t \mid \boldsymbol{\xi}, \mathcal{D}_{1:t-1}) \\ &= p(\mathbf{x}_t^r \mid \boldsymbol{\xi}, \mathcal{D}_{1:t-1}) \\ &= \mathcal{N}(\mathbf{x}_t^r \mid \hat{f}_{\boldsymbol{\xi}}(\mathbf{x}_{t-1}^r, \mathbf{u}_{t-1}), \mathbf{\Sigma}_{\text{obs}}), \end{aligned} \tag{4.16}$$

where we leverage the symmetry of the Gaussian distribution to centre the uncertainty around $\mathbf{x}_t^r$. It follows that, because the state transition is Markovian, the likelihood of $\ell(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$ depends only on the current observation tuple given by $\mathcal{D}_t$, and we can drop the conditioning on previously observed data. In other words, we now have a way to quantify how likely is a given realisation of $\boldsymbol{\xi}$ based on the data we have collected from the environment. Furthermore, let us define a single observation $\mathcal{D}_t = (\mathbf{x}_t^r, \mathbf{u}_{t-1}^r, \mathbf{x}_{t-1}^r)$ as the tuple of last applied control action and observed state transition. Inferring exclusively over the current state is useful whenever frequent observations are received and prevents us from having to store information over the entire observation dataset. This approach is also followed by ensemble Kalman filters and particle flow filters (Leeuwen et al., 2019).

Equipped with Eq. (4.16) and assuming that an initial prior $p(\boldsymbol{\xi})$ is available, we can proceed as discussed in Section 2.6 by approximating each prior at time t with a set of particles $\{\boldsymbol{\xi}^i\}_{i=1}^{N_\xi}$ following $q(\boldsymbol{\xi} \mid \mathcal{D}_{1:t-1})$, so that our posterior over $\boldsymbol{\xi}$ at a given time t can then be rewritten as:

$$p(\boldsymbol{\xi} \mid \mathcal{D}_{1:t}) \approx q(\boldsymbol{\xi} \mid \mathcal{D}_{1:t}) \propto \ell(\boldsymbol{\xi} \mid \mathcal{D}_t)\, q(\boldsymbol{\xi} \mid \mathcal{D}_{1:t-1}), \tag{4.17}$$

and we can make recursive updates to $q(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$ by employing it as the prior distribution for the following step. Namely, we can iteratively update $q(\boldsymbol{\xi} \mid \mathcal{D}_{1:t})$ a number of steps L by applying the update rule with a functional gradient computed as in Eq. (2.40), where:

$$\nabla_{\xi} \log p_t(\xi \mid \mathcal{D}_{1:t}) \propto \nabla_{\xi} \log p(\mathcal{D}_t \mid \xi) + \nabla_{\xi} \log q_t(\xi \mid \mathcal{D}_{1:t-1}). \tag{4.18}$$

An element needed to evaluate Eq. (4.18) is an expression for the gradient of the posterior density. An issue in sequential Bayesian inference is that there is no exact expression for the posterior density (Pulido & van Leeuwen, 2019). Namely, we know the likelihood function, but the prior density is only represented by a set of particles, not the density itself.

One could forge an empirical distribution $q(\xi \mid \mathcal{D}_{1:t}) = \frac{1}{N_\xi} \sum_{i=1}^{N_\xi} \delta(\xi^i)$ by assigning Dirac functions at each particle location, but we would still be unable to differentiate the posterior. In practice, we need to apply an efficient density estimation method, as we need to compute the density at each optimisation step. We choose to approximate the posterior density with an equal-weight Gaussian Mixture Model (GMM) with a fixed diagonal covariance matrix:

$$q(\xi \mid \mathcal{D}_{1:t}) = \frac{1}{N_\xi} \sum_{i=1}^{N_\xi} \mathcal{N}(\xi \mid \xi^i, \mathbf{\Sigma}_\mathrm{s}), \tag{4.19}$$

where the covariance matrix $\mathbf{\Sigma}_\mathrm{s}$ can be predetermined or computed from data. One option, for example, is to use a Kernel Density bandwidth estimation heuristic, such as Improved Sheather Jones (Botev, Grotowski & Kroese, 2010), Silverman's (Silverman, 1986) or Scott's (Scott, 1992) rule, to determine the standard deviation σ and set $\mathbf{\Sigma}_\mathrm{s} = \sigma^2 \mathbf{I}$.

One final consideration is that of the support for the prior distribution. Given the discussion above, it is important that the density approximation of $q(\xi \mid \mathcal{D}_{1:t})$ offers support on regions of interest in the parameter space. In our formulation, that can be controlled by adjusting $\mathbf{\Sigma}_\mathrm{s}$. Additionally, precautions have to be taken to make sure the parameter space is specified correctly, such as using log-transformations for strictly positive parameters for instance.

Algorithm 3: DuSt-MPC, Sim-to-Real in the loop SVMPC

```
   Input: $p(\xi), q(\theta_{t_0}), H, N_\xi, N_\pi, N_s, N_a, \alpha, \mathbf{\Sigma}_{obs}, \mathbf{\Sigma}_s, \mathbf{\Sigma}_a, \hat{f}_\xi(\cdot), C[\cdot]$
1  Sample $\{\theta^n_{t_0}\}^{N_\pi}_{n=1} \sim q(\theta_{t_0})$
2  foreach policy $\pi_{\theta^n} \in N_\pi$ do $\pi_{\theta^n} \leftarrow \mathcal{N}(\theta^n_{t_0}, \mathbf{\Sigma}_a)$
3  while task not complete do
4      $\mathbf{x}^r_t \leftarrow$ GetStateEstimate()
5      $\mathcal{D}_{1:t} \leftarrow \mathcal{D}_{1:t-1} \cup \{\mathbf{x}^r_t, \mathbf{x}^r_{t-1}, \mathbf{u}^r_{t-1}\}$
6      for $l \leftarrow 1$ to $L$ do in parallel
7          for $m \leftarrow 1$ to $N_\xi$ do
8              $\ell(\xi \mid \mathcal{D}_{1:t}) \leftarrow \mathcal{N}(\mathbf{x}^r_t \mid \hat{f}_\xi(\mathbf{x}^r_{t-1}, \mathbf{u}^r_{t-1}), \mathbf{\Sigma}_{obs})$
9              $\nabla_\xi \log p(\xi^m \mid \mathcal{D}_{1:t}) \approx \nabla_\xi \log \ell(\xi^m \mid \mathcal{D}_{1:t}) + \nabla_\xi \log q(\xi^m \mid \mathcal{D}_{1:t-1})$
10             $\phi(\xi^m) \leftarrow \frac{1}{N_\xi} \sum^{N_\xi}_{j=1} k(\xi^j, \xi^m) \nabla_{\xi^j} \log p_t(\xi^j \mid \mathcal{D}_{1:t}) + \nabla_{\xi^j} k(\xi^j, \xi^m)$
11             $\xi^m \leftarrow \xi^m + \epsilon \phi(\xi^m)$
12         $q(\xi \mid \mathcal{D}_{1:t}) = \frac{1}{N_\xi} \sum^{N_\xi}_{m=1} \mathcal{N}(\xi^m, \mathbf{\Sigma}_s)$
13     Sample $\{\xi_j\}^{N_s}_{j=1} \sim q(\xi \mid \mathcal{D}_{1:t})$
14     for $n \leftarrow 1$ to $N_\pi$ do in parallel
15         Sample $\{U^n_i\}^{N_a}_{i=1} \sim \pi_{\theta^n}$
16         foreach $i \in N_a$ and $j \in N_s$ do $C^n_{i,j} \leftarrow$ GetTrajCosts($U^n_i, \xi_j, \mathbf{x}^r_t$)
17         $\nabla_\theta \log p(\theta^n_t \mid \mathcal{O}, \xi) \approx \nabla_\theta \log q(\theta^n_{t-1}) + \nabla_\theta \log\Big(\frac{1}{N_a N_s} \sum^{N_a}_{i=1} \sum^{N_s}_{j=1} \exp(-\alpha C^n_{i,j})\Big)$
18         $\phi(\theta^n_t) \leftarrow \frac{1}{N_\pi} \sum^{N_\pi}_{i=1} k(\theta^i_t, \theta^n_t) \nabla_{\theta^i_t} \log p(\theta^i_t \mid \mathcal{O}, \xi) + \nabla_{\theta^i_t} k(\theta^i_t, \theta^n_t)$
19         $\theta^n_t \leftarrow \theta^n_t + \epsilon \phi(\theta^n_t)$
20         $\omega^n_t \leftarrow \frac{q(\theta^n_{t-1})}{N_a N_s} \sum^{N_a}_{i=1} \sum^{N_s}_{j=1} \exp(-\alpha C^n_{i,j})$
21     foreach policy $n \in N_\pi$ do $\omega^n_t \leftarrow \frac{\omega^n_t}{\sum^{N_\pi}_{n=1} \omega^n_t}$
22     $n^* = \arg\max_n \omega^n_t$
23     SendToActuators($U^{n^*}_t = \theta^{n^*}_t$)
24     $\theta_t \leftarrow$ RollPolicies($\theta_t$)
25     $q(\theta_t) = \sum^{N_\pi}_{n=1} \omega^n_t \mathcal{N}(\theta^n_t, \mathbf{\Sigma})$
26     foreach policy $n \in N_\pi$ do $\pi_{\theta^n} \leftarrow \mathcal{N}(\theta^n_t, \mathbf{\Sigma}_a)$
27     $t \leftarrow t + 1$
```

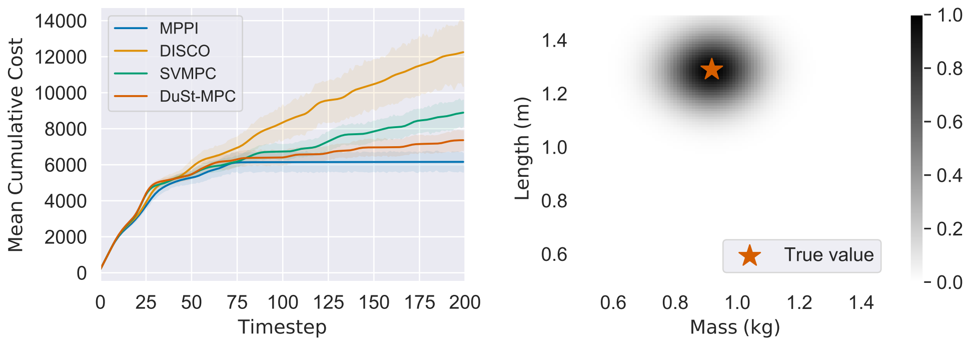


Figure 4.3: **Inverted pendulum**. The plots show the result of the experiment to balance an inverted pendulum with unknown pole-mass and length. **Left:** Mean cumulative cost over 10 episodes. The shaded region represents the 50% confidence interval. The high variance is expected since each scenario has parameters sampled from a uniform distribution. **Right:** Plot of the posterior distribution over the pendulum pole-mass and length at the final step of one of the episodes. The true latent value is shown by the red star marker.

4.6 Experiments

In the following section we present experiments, both in simulation and with a physical AGV, to demonstrate the correctness and applicability of our method. Although these experiments have low-dimensional control spaces it is important to note that in sampling-based MPC each control action in a sequential plan is independent. Therefore, the inference problem involves effectively solving a distribution whose dimensionality is the size of the control space *times* the number of control steps. As such, this demonstrates that the method is suitable in large-dimension spaces and that the control horizon can be adjusted according to the desired computational complexity.

For each experiment, the hyper-parameters α, ϵ, H, $\mathbf{\Sigma}$, $\mathbf{\Sigma}_\mathrm{a}$ where optimised by performing Bayesian Optimisation within a given search space (Akiba et al., 2019). The optimal values found where subsequently rounded-off to fewer significant digits and are listed in Appendix B.1.

4.6.1 Inverted pendulum with uncertain parameters

We first investigate the performance of DuSt-MPC in the classic inverted pendulum control problem. As usual, the pendulum is composed of a rigid pole-mass system controlled at one end by a 1-degree-of-freedom torque actuator. The task is to balance the point-mass upright, which, as the controller is typically under-actuated, requires a controlled swing motion to overcome gravity. Contrary to the typical case, however, in our experiments the mass and length of the pole are unknown and equally likely within a range of $0.6\,\mathrm{kg}$ to $1.3\,\mathrm{kg}$ and $0.6\,\mathrm{m}$ to $1.3\,\mathrm{m}$, respectively.

At each episode, a set of latent model parameters is sampled and used in the simulated environment. Each method is then deployed utilising this same parameter set. MPPI is used as a baseline and has *perfect knowledge* of the latent parameters. This provides a measure of the task difficulty and achievable results. As discussed in Section 4.1, we compare against DISCO and SVMPC as additional baselines. We argue that, although these methods perform no online update of their knowledge of the world, they offer a good underpinning for comparison since the former tries to leverage the model uncertainty to create more robust policies, whereas the latter shares the same variational inference principles as our method. DISCO is implemented in its unscented transform variant applied to the uninformative prior used to generate the random environments. SVMPC uses the mean values for mass and length as point-estimates for its fixed parametric model. For more details on the hyper-parameters used, refer to Appendix B.1.

Figure 4.3 presents the average cumulative costs over 10 episodes. Although the results show great variance, due to the randomised environment, it is clear that DuSt-MPC outperforms both baselines. Careful consideration will show that the improvement is more noticeable as the episode progresses, which is expected as the posterior distribution over the model parameters being used by DuSt-MPC gets more informative. Additionally, the final distribution over mass and length for one of the episodes can also be seen in Fig. 4.3. Finally, a summary of the experimental results is presented in Table 4.1.

4.6.2 Point-mass navigation on an obstacle grid

Here, we reproduce and extend the planar navigation task presented in Lambert et al. (2020). We construct a scenario in which an holonomic point-mass robot must reach a target location while avoiding obstacles. As in Lambert et al. (2020), colliding with obstacles not only incurs a high cost penalty to the controller, but prevents all

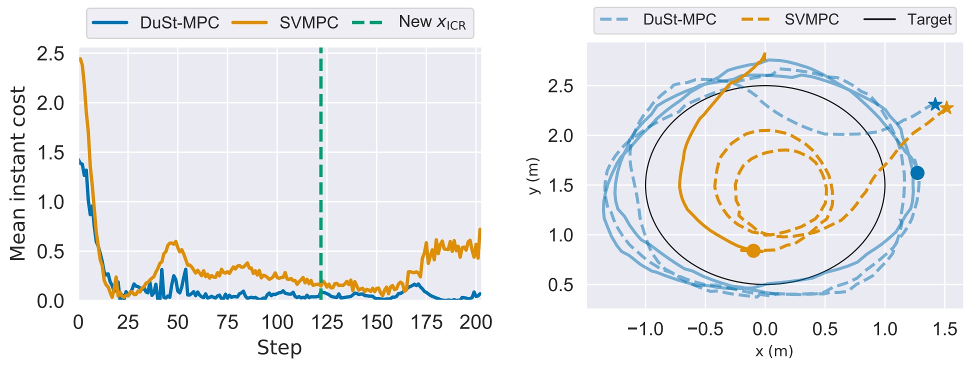


Figure 4.4: **AGV trajectory tracking**. The plots show the result of the Wombot trajectory tracking task with varying load. **Left:** Raw cost over time. Amount of steps before and after the change of mass are normalised for proper comparison. **Right:** Trajectories executed by each method. Line style changes when mass changes. Markers denote initial and change of mass position.

	Point-mass		Pendulum	
	Cost ($\mu \pm \sigma$)	Success[†]	Cost ($\mu \pm \sigma$)	Success[‡]
MPPI[§]	—	—	30.8±12.6	100%
DISCO	250.8±29.9	20%	61.3±40.0	70%
SVMPC	191.7±56.5	25%	44.5±17.9	70%
DuSt-MPC	118.3± 7.9	100%	36.8±14.0	80%

Table 4.1: **Simulation results**. Summary of results for simulation experiments. The mean episode cost is given by the sum of the instant costs over the episode length. Values shown do not include the crash penalty for a more comparable baseline. [§] Not used in the navigation task; has perfect knowledge in the pendulum task. [†] Successes are episodes with no crashes. [‡] Successes are episodes whose last five steps have a instant cost below 4 ($\approx 10°$ from the upright position).

future movement, simulating a crash. The non-differentiable cost function makes this a challenging problem, well-suited for sampling-based approaches. Obstacles lie in an equally spaced 4-by-4 grid, yielding several multi-modal solutions. This is depicted in Fig. 4.2. Additionally, we include barriers at the boundaries of the simulated space to prevent the robot from easily circumventing obstacles.

The system dynamics is represented as a double integrator model with a non-unitary mass m, s.t. the particle acceleration is given by $\ddot{\mathbf{x}} = m^{-1}\mathbf{u}$ and the control signal is the force applied to the point-mass. In order to demonstrate the inference over dynamics, we forcibly change the mass of the robot at a fixed step of the experiment, adding extra weight. This has a direct parallel to several tasks in reality, such as collecting a payload or passengers while executing a task. Assuming the goal position is denoted by $\mathbf{x}_g$, the cost function that defines the task is given by:

$$\begin{aligned} \mathcal{C}_{\text{inst}}(\mathbf{x}_t, \mathbf{u}_t) &= 0.5\mathbf{e}_t^\intercal \mathbf{e}_t + 0.25\dot{\mathbf{x}}_t{}^\intercal \dot{\mathbf{x}}_t + 0.2\mathbf{u}_t^\intercal \mathbf{u}_t + p \cdot \mathbb{1}\{\text{col.}\} \\ \mathcal{C}_{\text{term}}(\mathbf{x}_t, \mathbf{u}_t) &= 1000\mathbf{e}_t^\intercal \mathbf{e}_t + 0.1\dot{\mathbf{x}}_t{}^\intercal \dot{\mathbf{x}}_t, \end{aligned}$$

where $\mathbf{e}_t = \mathbf{x}_t - \mathbf{x}_g$ is the instantaneous position error and $p = 10^6$ is the penalty when a collision happens. A detailed account of the hyper-parameters used in the experiment is presented in Appendix B.1.

As a baseline, we once more compare against DISCO and SVMPC. In Fig. 4.2 we present an overlay of the trajectories for SVMPC and DuSt-MPC over 20 independent episodes and we choose to omit trajectories of DISCO for conciseness. Collisions to obstacles are denoted by a **x** marker. Note that in a third of the episodes SVMPC is unable to avoid obstacles due to the high model mismatch while DuSt-MPC is able to avoid collisions by quickly adjusting to the new model configuration online. A typical sequential plot of the posterior distribution induced by fitting a GMM as in Eq. (4.19) is shown on Fig. 4.2c for one episode. There is little variation between episodes and the distribution remains stable in the intermediate steps not depicted.

4.6.3 Trajectory tracking with autonomous ground vehicle

We now present the experimental results with a physical AGV equipped with a skid-steering drive mechanism. The kinematics of the robot are based on a modified unicycle model, which accounts for skidding via an additional parameter (Kozłowski & Pazderski, 2004). The parameters of interest in this model are the robot's wheel radius r_w, axial distance a_w, i.e. the distance between the wheels, and the displacement of the robot's ICR from the robot's centre x_{ICR}. A non-zero value on the latter affects turning by sliding the robot sideways. The robot is velocity controlled and, although it possess four-wheel drive, the controls is restricted to two-degrees of freedom, left and right wheel speed. Individual wheel speeds are regulated by a low-level proportional-integral controller.

The robot is equipped with a 2-D Hokuyo LiDAR and operates in an indoor environment in our experiments. Prior to the tests, the area is pre-mapped using the *gmapping* package (Grisetti, Stachniss & Burgard, 2007) and the robot is localised against this pre-built map. Similar to the experiment in Section 4.6.2, we simulate a change in the environment that could be captured by our parametric model of the robot to explain the real trajectories. However, we are only applying a relatively simple kinematic model in which the effects of the dynamics and ground-wheel interactions are not accounted for. Therefore, friction and mass are not feasible inference choices. Hence, out of the available parameters, we opted for inferring x_{ICR}, the robot's centre of rotation. Since measuring x_{ICR} involves a laborious process, requiring different weight measurements or many trajectories from the physical hardware (Yi et al., 2009), this also makes the experiment more realistic. To circumvent the difficulties of ascertaining x_{ICR}, we use the posterior distribution estimated in Barcelos et al. (2020), and bootstrap our experiment with $x_{\text{ICR}} \sim \mathcal{N}(0.5, 0.2^2)$.

To reduce the influence of external effects, such as localisation, we defined a simple control task of following a circular path at a constant tangential speed. Costs were set to make the robot follow a circle of $1\,\text{m}$ radius with $C_{\text{inst}}(\mathbf{x}_t) = \sqrt{d_t^2 + 10(s_t - s_0)^2}$, where d_t represents the robot's distance to the edge of the circle and $s_0 = 0.2\,\text{m}\,\text{s}^{-1}$ is a reference linear speed.

The initial particles needed by DuSt-MPC in Eq. (4.18) for the estimation of x_{ICR} are sampled from the bootstrapping distribution, whereas for SVMPC we set $x_{\text{ICR}} = 0.5\,\text{m}$, the distribution mean. Again, we want to capture whether our method is capable of adjusting to environmental changes. To this end, approximately halfway through the experiment, we add an extra load of approximately $5.3\,\text{kg}$ at the rear of the robot in order to alter its centre of mass. These moments are indicated on the trajectories shown in Fig. 4.4. Also in Fig. 4.4 we plot the instant costs for a fixed number of steps before and after the change of mass. For the complete experiment parameters refer to the Appendix B.1.

We observe that considering the uncertainty over x_{ICR} and, crucially, refining our knowledge over it (see Fig. 4.1), allows DuSt-MPC to significantly outperform the SVMPC baseline. Focusing on the trajectories from SVMPC we note that our estimation of x_{ICR} is probably not accurate. As the cost function emphasises the tangential speed over the cross-track error, this results in circles correctly centred, but of smaller radius. Crucially though, the algorithm cannot overcome this poor initial estimation. DuSt-MPC initially appears to find the same solution, but quickly adapts, overshooting the target trajectory and eventually converging to a better result. This behaviour can be observed both prior to and after the change in the robot's mass. Conversely, with the addition of mass, the trajectory of SVMPC diverged and eventually led the robot to a halt.

4.7 Summary

In Chapter 3 we have seen how incorporating model uncertainty in a control problem can be challenging, but lead to improved performance, especially in the presence of disturbances or unmodelled phenomena. One of the drawbacks of the first approach was the fact that the uncertainty propagation was accurate up to certain momenta of the posterior distribution and uni-modal. On the other hand, SVMPC offers a way to deal with multi-modal inference, but does not take model uncertainty into account.

In this chapter we presented DuSt-MPC, a method to perform multi-modal inference under model uncertainty. Additionally, using a particle filtering inspired approach, DuSt-MPC is able to encapsulate gathered observations into a prior distribution over model parameters and refine such distribution in real-time. This provides a formal way of adapting the control policy to changes in the environment, such as covariate shift, while keeping a Bayesian viewpoint on previously observed situations. The experimental results illustrate how this method is more efficient than both DISCO and SVMPC in handling problems with parametric uncertainty, such as the pendulum experiment, and unforeseen disturbances, such as the particle maze navigation.

However, both SVMPC and DuSt-MPC may suffer from lack of diversity once the dimensionality of the inference problem is large. This problem is further compounded by the fact that the similarity of trajectories is computed over flattened vector spaces of dimension $\mathbb{R}^{m \times H}$, where m are the dimensions of the control space and H the amount of steps in the control horizon. In other words, for controllers with a long look-ahead or several degrees of freedom, the problem dimension can quickly grow. This poses a significant challenge for particle-based variational inference, as the repulsive forces imposed by the kernel gradient become smaller as the number of dimensions increase (Zhuo et al., 2018). The alternative proposed by Zhuo et al. (2018) is to compute the embedding for each step of the control policy separately and aggregate them later through a graph-based message passing kernel. This, however, creates an auto-regressive kernel which has a substantially larger computational burden.

In the next chapter we will investigate an alternative way to tackle this issue by exploring how trajectories can be represented differently and computing their similarities in a projected feature space. This approach allows us to leverage desired properties of trajectories and promotes higher particle diversity on the approximated posterior distribution, which in turn lead to optimisation solution with better global properties.

In diversity there is beauty and there is strength.

Maya Angelou

CHAPTER 5

Improving solution diversity with path signatures

On the previous chapter we have seen how framing an MPC problem as an equivalent inference problem allows us to reason about multi-modal solutions and take into account model uncertainty. We have introduced a VI approach named DuSt-MPC that is able to approximate a posterior policy distribution while updating a distribution over possible simulator parameters. This is achieved either by Monte Carlo sampling, when the simulator and/or cost function is not differentiable, or otherwise by direct optimisation. Conversely, we have discussed how this inference problem can unravel into a high-dimensional solution space depending on the control horizon, which makes the problem significantly harder to solve. We posit this is due to two main reasons: first, the sampling space becomes larger and hence we need more Monte

Carlo samples to achieve reasonable coverage; and second, the repulsive force between particles diminishes substantially, leading to a concentration of solutions around local minima.

In this chapter we will introduce the use of *Path Signatures* as an alternative representation of trajectories over arbitrary spaces. We will show how this representation functions as a canonical feature map that is able to promote higher particle diversity and leads to a posterior policy distribution with better global properties. This is demonstrated not only on applied control problems, as those illustrated on the previous chapters, but also on problems of trajectory optimisation which offer an easier way to visualise solutions.

Trajectory optimisation is one of the key tools in robotic motion, used to find control signals or paths in obstacle-cluttered environments that allow the robot to perform desired tasks. These trajectories can represent a variety of applications, such as the motion of autonomous vehicles or robotic manipulators. In most problems, we consider a *state-space model*, where each distinct situation for the world is called a *state*, and the set of all possible states is called the *state space* (LaValle, 2006). When optimising candidate trajectories for planning and control, two criteria are usually considered: *optimality* and *feasibility*. Although problem dependant, in general, the latter evaluates in a binary fashion whether the paths generated respect the constraints of both the robot and the task, such as physical limits and obstacle avoidance. Conversely, optimality is a way to measure the quality of the generated trajectories with respect to task-specific desired behaviours. For example, if we are interested in smooth paths we will search for trajectories that minimise changes in velocity and/or acceleration. The complexity of most realistic robot planning problems scales exponentially with the dimensionality of the state space and is countably infinite. When focusing on motion planning, a variety of algorithms have been proposed to find optimal and feasible trajectories. These can be roughly divided into two main paradigms: sampling-based and trajectory optimisation algorithms.

Sampling-based planning (Gammell & Strub, 2021) is a class of planners with *probabilistically complete* and *asymptotically optimal* guarantees (Al-Bluwi, Siméon & Cortés, 2012). These approaches decompose the planning problem into a series of sequential decision-making problems with a tree-based (LaValle & Kuffner, 2001) or graph-based (Kavraki et al., Aug./1996) approach. However, they are limited in their ability to encode kinodynamic cost like trajectory curvature (Heilmeier et al., 2020) or acceleration torque limits (Berntorp et al., 2014). In addition, despite the completeness guarantee, sampling-based planners are often more computationally expensive as the search space grows and can obtain highly varying results due to the random nature of the algorithms.

Trajectory optimisation algorithms (Gonzalez et al., 2016) use a diverse set of techniques to minimise a cost functional that encourages solutions to be both optimal and feasible. The most direct optimisation procedure relies on a differentiable cost function and uses functional gradient techniques to iteratively improve the trajectory quality (Ratliff et al., 2009). However, many different strategies have been proposed. For example, one may start from a randomly initialised candidate trajectory and proceed by adding random perturbations to explore the search space and generate approximate gradients, allowing any arbitrary form of cost functional to be encoded (Kalakrishnan et al., 2011). The same approach can be used to search for control signals and a local motion plan concurrently (G. Williams et al., 2016). Finally, a locally optimal trajectory can also be obtained via decomposing the planning problem with sequential quadratic programming (Schulman et al., 2013). A drawback of these methods is that they usually find solutions that are locally optimal and may need to be run with different initial conditions to find solutions that are feasible or with lower costs.

In the present work we will focus on trajectory optimisation. More specifically, in a class of algorithms that performs parallel optimisation of a batch of trajectories. The concurrent optimisation of several paths in itself already alleviates the proneness to local solutions. Nonetheless, we show how a proper representation of trajectories when performing functional optimisation leads to increased diversity and solutions with a better global property, either with direct gradients or Monte Carlo-based gradient approximations. As an illustrative example, refer to Fig. 5.2.

Our approach is based on two cornerstones. On one hand, we use a modification of SVGD (Liu & Wang, 2016), a variational inference method to approximate a posterior distribution with an empirical distribution of sampled particles, to optimise trajectories directly on a structured RKHS. The structure of this space is provided by the second pillar of our approach. We leverage recent advancements in rough path theory to encode the sequential nature of paths in the RKHS using a Path Signature Kernel (Kiraly & Oberhauser, 2019; Salvi et al., 2021). Therefore we can approximate

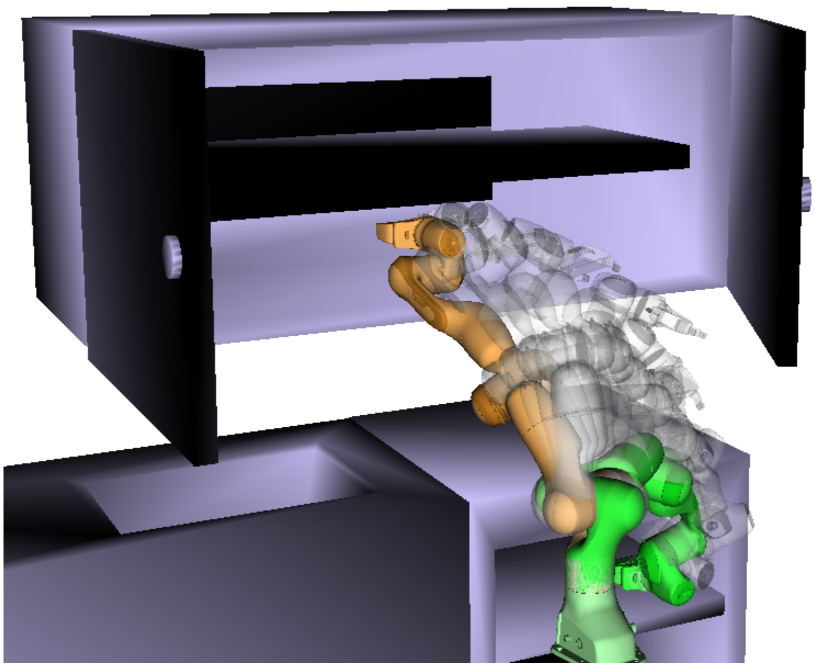


Figure 5.1: **An episode of the *Kitchen* scene.** Depicted is one of the collision-free paths found by Kernel Signature Variational Gradient Descent (SigSVGD) on a reaching task using a 7 Degrees of Freedom (DOF) Franka Panda arm on the MotionBenchMaker planning benchmark.

the posterior distribution over optimal trajectories with structured particles during the optimisation while still taking into account motion planning and control idiosyncrasies. More concretely, the main contributions of this chapter are: we introduce the use of path signatures (T. Lyons, 2014) as a canonical feature map to represent trajectories over high-dimensional state spaces; next, we outline a procedure to incorporate the signature kernel into a variational inference framework for motion planning; finally, we demonstrate through experiments in both planning and control that the proposed procedure results in more diverse trajectories, which aid in avoiding local minima and lead to a better optimisation outcome.

The chapter is organised as follows. In Section 5.1 we review related work, contrasting the proposed method to the existing literature. In Section 5.2 we provide background on path signatures and motion planning as variational inference, which are the foundational knowledge for the method outlined in Section 5.3. Finally, in Section 5.4 we present a number of simulated experiments, followed by relevant discus-

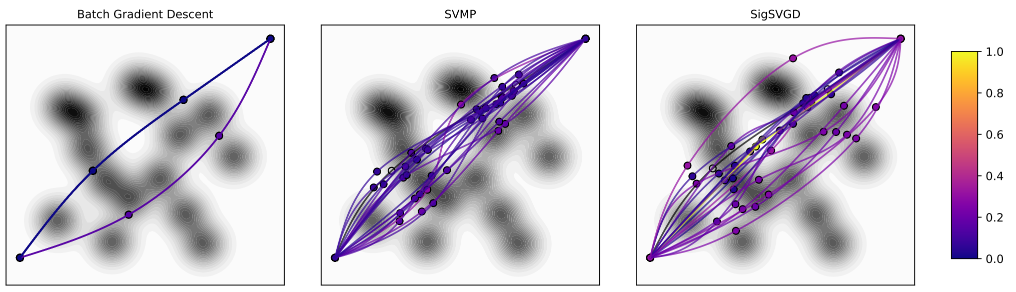


Figure 5.2: **Qualitative analysis of 2-D planning task**. The plot shows the final 20 trajectories found with different optimisation methods. The colour of each path shows its normalised final cost. Note how all batch gradient descent trajectories converge to two modes of similar cost. Paths found by SVMP are already more diverse, but one of the gradient descent modes is lost. Note how when multiple trajectories converge to a single trough, the knots are pushed away by the repulsive force resulting in suboptimal solutions. Conversely, paths found by SigSVGD are diverse and able to find more homotopic solutions, including those found by BGD. Note also how paths are able to converge to the same trough without being repelled by one another since the repulsive force takes into account the entire trajectory and not exclusively the spline knot placement. That also allows for paths that are more direct and coordinated than SVMP.

sions in Section 5.5.

5.1 Related work

Trajectory optimisation refers to a class of algorithms that start from an initial sub-optimal path and find a, possibly local, optimal solution by minimising a cost function. Given its broad definition, there are many seminal works in the area. One influential early work is Covariant Hamiltonian Optimisation for Motion Planning (CHOMP) (Ratliff et al., 2009) and related methods (Byravan et al., 2014; Marinho et al., 2016; Zucker et al., 2013). The algorithm leverages the covariance of trajectories coupled with Hamiltonian Monte Carlo to perform annealed functional gradient descent. However, one of the limitations of CHOMP and related approaches is the need for a fully-differentiable cost function.

In Stochastic Trajectory Optimisation for Motion Planning (STOMP) (Kalakrishnan et al., 2011) the authors address this by approximating the gradient from stochastic samples of noisy trajectories, allowing for non-differentiable costs. Another approach used in motion planning are quality diversity algorithms, at the intersection of optimisation and evolutionary strategies, of which Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is the most prominent (Hämäläinen et al., 2020; Hansen, Müller & Koumoutsakos, 2003; Tjanaka et al., 2022). CMA-ES is a derivative-free method that uses a multivariate normal distribution to generate and update a set of candidate solutions, called individuals. The algorithm adapts the covariance matrix of the distribution based on the observed fitness values of the individuals and the search history, balancing exploration and exploitation of the search space. Because of its stochastic nature, it is ergodic and copes well with multi-modal problems. Nonetheless, it may require multiple initialisations and it typically requires more evaluations than gradient-based optimisers (Hansen, 2016).

TrajOpt (Schulman et al., 2013), another prominent planner, adopts a different approach solving a sequential quadratic program and performing continuous-time collision checking. Contrary to sampling-based planners, these trajectory optimisation methods are fast, but only find locally optimal solutions and may require reiterations until a feasible solution is found. Another issue common to these approaches is that in practice they require a fixed and fine parametrisation of trajectory waypoints to ensure feasibility and smoothness, which negates the benefit of working on continuous trajectory space. To address this constraint, in (Marinho et al., 2016) the authors restrict the optimisation and trajectory projection to an RKHS with an associated squared-exponential kernel. However, the cost between sparse waypoints is ignored and the search is still restricted to a deterministic trajectory. Another approach was proposed in Gaussian Process Motion Planing (GPMP) (Dong et al., 2016; Mukadam et al., 2017, 2018) by representing trajectories as GPs and looking for a *maximum a posteriori* (MAP) solution of the inference problem.

More closely related to our approach are (Lambert & Boots, 2021; Yu & Chen, 2022) which frame motion planning as a variational inference problem and try to estimate the posterior distribution represented as a set of trajectories. In (Yu & Chen, 2022), the authors modify GPMP with a natural gradient update rule to approximate the posterior. On the other hand, in Stein Variational Motion Planning (SVMP) (Lambert & Boots, 2021) the posterior inference is optimised using Stein variational gradient descent. This method is similar to ours, but the induced RKHS does not take into account the sequential nature of the paths being represented, which leads to a diminished repulsive force and lack of coordination along the dimensions of the projected space.

In contrast, our approach—which we will refer to as SigSVGD—uses the path signature to encode the sequential nature of the functional being optimised. We argue that this approach leads to a better representation of trajectories promoting diversity and finding better local solutions. To empirically corroborate this claim we use the Occam's razor principle and take SVMP as the main baseline of comparison since it more closely approximates our method.

We note that the application of trajectory optimisation need not be restricted to motion planning. By removing the constraint of a target state and making the optimisation process iterative over a rolling horizon we retrieve a wide class of Model Predictive Controllers with applications in robotics (Barcelos et al., 2020, 2021; Lambert et al., 2020; G. Williams et al., 2016). SVMPC (Lambert et al., 2020) uses variational inference with SVGD optimisation to approximate a posterior over control policies and more closely resembles SigSVGD. However, like SVMP, it too does not take into account the sequential nature of control trajectories and we will illustrate how our approach can improve the sampling of the control space and promote better policies.

5.2 Background

5.2.1 Trajectory optimisation for planning and control

Consider a system with state $\mathbf{x} \in \mathcal{X}$ and let us denote a *trajectory* of such system as $X : [a, b] \to \mathcal{X}$, where $\mathcal{X}$ is an appropriate Euclidean space or group. We shall use the notation X_t to denote the dependency on time $t \in [a, b]$. The trajectory X describes a *path* in $\mathcal{X}$ and we shall use the two denominations interchangeably. In trajectory optimisation the goal is to find the optimal path X^* from a given starting state $\mathbf{x}_s$ to a certain goal state $\mathbf{x}_g$. This can be done by minimising a cost functional that codifies our desired behaviour $\mathcal{C} : \mathcal{P}_{\mathcal{X}} \to \mathbb{R}^+$, where $\mathcal{P}_{\mathcal{X}}$ is the Hilbert space of trajectories (King et al., 2013):

$$\begin{aligned} X^* := \arg\min_X \quad & \mathcal{C}(X) \\ \text{s.t.} \quad & X_a = \mathbf{x}_s \\ & X_b = \mathbf{x}_g. \end{aligned} \tag{5.1}$$

In the typical case $\mathcal{C}$ is a bespoke functional that includes penalties for trajectory non-smoothness, total energy, speed and acceleration tracking, as well as length. To ensure that the solution is feasible and collision-free, additional equality and inequality constraints may also be included (Schulman et al., 2013). Alternatively, we can solve an unconstrained problem and include additional penalties to the cost functional as soft-constraints (Ratliff et al., 2009; Zucker et al., 2013).

Finally, note that the problem stated in Eq. (5.1) can be viewed as an open-loop optimal control problem. If the solution can be found in a timely manner, the same problem can be cast onto a Model Predictive Control (Barcelos et al., 2020, 2021; Camacho & Alba, 2013) framework

$$\begin{aligned} U^* := \arg\min_U \quad & \mathcal{C}(X, U) \\ \text{s.t.} \quad & X_a = \mathbf{x}_s, \end{aligned} \tag{5.2}$$

where $U : [a, b] \to \mathcal{U}$ is a path of control inputs on a given Euclidean space and the mapping to $\mathcal{X}$ is given by the dynamical system f such that $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$. That is to say, we now influence the path X indirectly through input U, and at any time t the problem is solved for a finite interval. The closed-loop solution arises from applying only the first immediate control action before re-optimising the solution.

5.3 Method

Our main goal is to find a diverse set of solutions to the problem presented in Section 5.2.1. To that end, we begin by reformulating Eq. (5.1) as a probabilistic inference problem. Next, we show that we can apply SVGD to approximate the posterior distribution of trajectories with a set of sampled paths. Finally, in Section 5.3.3, we present our main contribution discussing how we can promote diversity among the sample paths by leveraging the Path Signature Kernel.

5.3.1 Stein Variational Motion Planning

To reframe the trajectory optimisation problem described in Eq. (5.1) as probabilistic inference we introduce a binary optimality criterion, $\mathcal{O} : \mathcal{P}_{\mathcal{X}} \to \{0, 1\}$, analogously to (Barcelos et al., 2021; Levine, 2018). Simplifying the notation with $\mathcal{O}$ indicating $\mathcal{O} = 1$, we can represent the posterior distribution of optimal trajectories as $p(X \mid \mathcal{O}) \propto p(\mathcal{O} \mid X)\, p(X)$, for a given optimality likelihood $p(\mathcal{O} \mid X)$ and trajectory prior $p(X)$. The MAP solution is given by finding the mode of the negative log posterior:

$$\begin{aligned} X^* &= \arg\min_X - \log p(\mathcal{O} \mid X) - \log p(X) \\ &= \arg\min_X \lambda\, C(X) - \log p(X), \end{aligned} \tag{5.3}$$

where the last equality arises from the typical choice of the exponential distribution to represent the optimality likelihood, i.e. $p(\mathcal{O} \mid X) = \exp(-\lambda\, C(X))$ with λ being a temperature hyper-parameter.

Rather than finding the MAP solution, we are interested in approximating the full posterior distribution, which may be multi-modal, and generating diverse solutions for the planning problem. As discussed in Section 2.6, we can apply SVGD to approximate the posterior distribution with a collection of particles. In the case at hand each of such particles is a sampled path, such that Eq. (2.40) can be rewritten as:

$$\boldsymbol{\phi}^*(X) = \mathbb{E}_{Y \sim \hat{q}}\big[k(Y, X)\, \nabla_Y \log p(Y \mid \mathcal{O}) + \nabla_Y k(Y, X)\big]. \tag{5.4}$$

The score function presented in Eq. (5.4) is composed of two competing forces. On one hand, we have the kernel smoothed gradient of the log-posterior pushing particles towards regions of higher probability. Whereas the second term acts as a repulsive force, pushing particles away from one another.

It is worth emphasising that the kernel function is *static*, i.e. it does not consider the sequential nature of the input paths. In effect, for a path of dimension c and x discrete time steps, the inputs are projected onto a space $\mathcal{V} \subset \mathbb{R}^{c \times x}$ in which similarities are evaluated.

Finally, the posterior gradient can be computed by applying Bayes' rule, resulting in:

$$\nabla_Y \log p(Y \mid \mathcal{O}) = \nabla_X \log p(Y) - \nabla_Y \lambda\, C(Y). \tag{5.5}$$

5.3.2 Stein Variational Motion Planning with smooth paths

In previous work (Barfoot, Hay Tong & Sarkka, 2014; Dong et al., 2016; Lambert & Boots, 2021; Mukadam et al., 2018) the prior distribution in Eq. (5.5) is defined in a way to promote smoothness on generated paths. This typically revolves around defining Gaussian Processes (Rasmussen & Williams, 2006) as priors and leveraging factor graphs for efficiency. Although effective, this approach still requires several latent variables to describe a desired trajectory, which implies on a higher dimensional inference problem.

Importantly, the problem dimensionality is directly related to the amount of repulsive force exerted by the kernel. In large dimensional problems, the repulsive force of translation-invariant kernels vanishes, allowing particles to concentrate around the posterior modes which results in an underestimation of the posterior variance (Zhuo et al., 2018). This problem is further accentuated when considering the static nature of the kernel function, as discussed in the previous section.

In order to keep the inference problem low-dimensional while still ensuring paths are smooth we make use of *natural cubic splines* and aim to optimise the location of a small number of knots. These knots may be initialised in different ways, such as perturbations around a linear interpolation from the starting state $\mathbf{x}_s$ and goal state $\mathbf{x}_g$, sampled from an initial solution given by a shooting method—e.g. Rapidly-exploring Random Tree (RRT) (LaValle & Kuffner, 2001)—or drawn randomly from within the limits of $\mathcal{X}$. For simplicity, we will opt for the latter.

Since path smoothness is induced by the splines, the choice of prior is more functionally related to the problem at hand. If one desires some degree of regularisation on the trajectory optimisation, a multivariate Gaussian prior centred at the placement of the initial knots may be used. Conversely, if we only wish to ensure the knots are within certain bounds, a less informative smoothed approximation of the uniform prior may be used. More concretely, for a box $B = x : \; a \leq x \leq b$, such prior would be defined as:

$$p(x) \propto \exp\left(-d(x,B)^2/\sqrt{(2\sigma^2)}\right) \tag{5.6}$$

where the distance function $d(x, B)$ is given by $d(x, B) = \min |x - x'|, \; x' \in B$. Finally, we could define both a prior and hyper-prior if we wish to combine both effects (see Appendix C.3 for details).

As discussed in Section 5.2.1 the cost functional C imposes penalties for collisions and defines the relevant performance criteria to be observed. Since only a small number of knots is used for each path, some of these criteria and, in particular, collision checking require that we discretise the resulting spline in a sufficiently dense amount of points. It is worth mentioning that C is typically non-differentiable and that the gradient in Eq. (5.5) is usually approximated with Monte Carlo samples (Barcelos et al., 2021). However, as this introduces an extra degree of stochasticity in the benchmark comparison, we will restrict our choice of C to be differentiable. We will discuss the performance criteria of each problem in the experimental section.

5.3.3 Stein Variational Motion Planning with path signature kernel

In this section we present our main contribution, which is a new formulation for motion planning in which Path Signature can be used to efficiently promote diversity in trajectory optimisation through the use of Signature Kernels. In other words, we want to define a kernel $k^+ : \mathcal{P}_\mathcal{X} \times \mathcal{P}_\mathcal{X} \to \mathbb{R}$, which takes into account the structure induced by paths, instead of the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined by SVMP. In Section 5.2 we discussed some desirable properties of the signature transform. The key insight is that the space of linear combination of signatures forms an algebra, which enables it as a faithful feature map for trajectories (Kiraly & Oberhauser, 2019).

With that in mind, perhaps the most straightforward use of the signature would be to redefine the kernel used in Eqs. (2.39) and (2.40) as $\bar{k}(X, Y) = k\big(\mathrm{S}(X)_s, \mathrm{S}(Y)_t\big)$, where we use the notation s and t to denote the independence between paths. However, as seen in Section 5.2, this approach would not be scalable given the exponential time and space complexity of the signature w.r.t. to its degree. A single evaluation of the Gram kernel matrix for $\bar{k}$ would be an operation of order $O(n^2 \cdot c^d)$, where n is the number of concurrent paths being optimised, d is the degree of the signature, and c is the dimensionality of the space $\mathcal{P}_\mathcal{X} \ni X, Y$.

Hence, we take a different approach and proceed by first projecting paths to an RKHS onto which we will then compute the signature. That is, given a kernel

$$k^+ : \mathcal{P}_\mathcal{X} \times \mathcal{P}_\mathcal{X} \to \mathbb{R}, \tag{5.7}$$

a path $X \in \mathcal{P}_{\mathcal{X}}$ can be lifted to a path in the RKHS $\mathcal{P}_{\mathcal{H}}$ through the map

$$k_X : s \mapsto k(X_s, \cdot), \tag{5.8}$$

where $\mathcal{P}_{\mathcal{H}}$ is the set of $\mathcal{H}$-valued paths. Finally, we compute the signature of the lifted path $\mathrm{S}(k_X)_s$ and use it as our final feature map.

At first glance, this further deteriorates scalability, since most useful $\mathcal{P}_{\mathcal{H}}$ are infinite dimensional, rendering this approach infeasible. However, results presented by Kiraly and Oberhauser (2019, Corollary 4.9) show that this approach can be completely kernelised. This allows them to define a *truncated signature kernel*,

$$k^{+} : (X_s, Y_t) \mapsto \left\langle \mathrm{S}^d(k_X)_s, \mathrm{S}^d(k_Y)_t \right\rangle, \tag{5.9}$$

that can be efficiently computed using only evaluations of a static kernel $k(X, Y)$ at discretised timestamps. The number of evaluations depends on the truncation degree d and number of discretised steps l. Several algorithmic approaches are considered in (Kiraly & Oberhauser, 2019) with dynamic programming having complexity $O(n^2 \cdot l^2 \cdot d)$ to compute a $(n \times n)$-Gram matrix. Otherwise, approximations can be used to reduce the complexity to linear on l and n. However, even though the importance of the terms in the signature decay factorially (T. Lyons, 2014), the amount of coefficients grows exponentially, which means that for high values of d the kernel k^{+} would be restricted to low-dimensional applications.

Nonetheless, recent work by Salvi et al. (2021) proved that for two continuously differentiable input paths the complete *signature kernel*,

$$k^{\oplus} : (X_s, Y_t) \mapsto \langle \mathrm{S}(k_X)_s, \mathrm{S}(k_Y)_t \rangle, \tag{5.10}$$

is also the solution of a second-order, hyperbolic Partial Differential Equation (PDE) known as Goursat PDE. More precisely, for two differentiable paths X and Y, the equivalent PDE equation is given by:

$$\begin{aligned} \frac{\partial^2 k^{\oplus}_{X,Y}}{\partial s \partial t} &= \langle \dot{k_X}, \dot{k_Y} \rangle_{\mathcal{P}_{\mathcal{H}}} k^{\oplus}_{X,Y} \\ \text{s.t.} \quad k^{\oplus}_{X,Y}(u, \cdot) &= k^{\oplus}_{X,Y}(\cdot, v) = 1, \end{aligned} \tag{5.11}$$

where k_X and k_Y are the projections of paths X and Y when lifted by the static kernel k; $\dot{k_X}$ and $\dot{k_Y}$ are the derivatives of the lifted paths; and $k^{\oplus}_{X,Y}(u, \cdot) = k^{\oplus}_{X,Y}(\cdot, v) = 1$ are the boundary conditions.

Salvi et al. (2021) show how Eq. (5.11) is an instance of a Goursat problem and how it can be numerically solved by finite differences approximation on a discretised grid along s and t. The choice of the discretisation grid offers a trade-off: the finer the grid, the better the approximation; but at the expense of a greater computational cost. To provide the reader with an intuition of how this approximation is constructed, consider that when using a first order approximation of the derivatives of $\dot{k_X}$ and $\dot{k_Y}$, Eq. (5.11) can be rewritten as:

$$\begin{aligned}\frac{\partial^2 k^{\oplus}_{X,Y}}{\partial s \partial t} &= \left(\langle k_{X_s}, k_{Y_t}\rangle - \langle k_{X_{s-1}}, k_{Y_t}\rangle - \langle k_{X_s}, k_{Y_{t-1}}\rangle + \langle k_{X_{s-1}}, k_{Y_{t-1}}\rangle\right) k^{\oplus}_{X,Y} \\ &= \left(k(X_s, Y_t) - k(X_{s-1}, Y_t) - k(X_s, Y_{t-1}) + k(X_{s-1}, Y_{t-1})\right) k^{\oplus}_{X,Y},\end{aligned} \tag{5.12}$$

which depends exclusively on computations of the static kernel and therefore circumvent the need to evaluate the signature of the path.

Solving this PDE is a problem of complexity $\mathcal{O}(l^2 \cdot c)$, so still restrictive on the discretisation of the path. However, by its intrinsic nature, the PDE can be parallelised, turning the complexity into $\mathcal{O}(l \cdot c)$, as long as the GPU is able to accommodate the required number of threads. Therefore the untruncated signature kernel can be efficiently and parallel computed using state-of-the-art hyperbolic PDE solvers and finite-difference evaluations of the static kernel k.

Hence, we can directly apply $k^{\oplus}$ in Eq. (5.4) and we now have a way to properly represent sequential data in feature space, resulting in the final gradient update function:

$$\boldsymbol{\phi}^*(X) = \mathbb{E}_{Y \sim \hat{q}}\left[k^{\oplus}(Y, X)\, \nabla_Y \log p(Y \mid \mathcal{O}) + \nabla_Y k^{\oplus}(Y, X)\right]. \tag{5.13}$$

For convenience, we will use the acronym SigSVGD whether the algorithm is used for planning or control problems. A complete overview of the algorithm is presented in Algorithm 4.

5.4 Results

In this section we present results to demonstrate the correctness and applicability of our method in a set of simulated experiments, ranging from simple 2-D motion planning to a challenging benchmark for robotic manipulators.

Algorithm 4: Kernel Signature Variational Gradient Descent

```
Input: A cost function $C(X)$ or target distribution $p(X)$, a prior
       distribution $q(X_{t_0})$, a signature kernel $k^{\oplus}$.
Output: A set of particles $\{X_t^i\}_{i=1}^{N_p}$ that approximates the posterior
        distribution over optimal paths.
1  Sample $\{X_{t_0}^i\}_{i=1}^{N_p} \sim q(X_{t_0})$;
2  while task not complete do
3      if using Monte Carlo samples then
4          Generate $N$ samples for each path $X_t^{i,jx} \leftarrow X_t^i + \eta_{jx}$;
5      if using splines then
6          Generate decimated trajectories from knots $X_t$;
7      Evaluate $C(X_t)$ in parallel;
8      if target distribution $p(X_t)$ is available then
9          Update score
           $\phi^* \leftarrow \frac{1}{N_p} \sum_i^{N_p} [k^{\oplus}(X_t^i, X_t)\, \nabla_{X_t^i} \log p(X_t^i) + \nabla_{X_t^i} k^{\oplus}(X_t^i, X_t)]$;
10     else
11         Log-posterior gradient $\nabla_{X_t^i} \log p(X_t^i \mid \mathcal{O}) \approx$
           $\nabla_{X_t^i} \log q(X_{t-1}^i \mid \mathcal{O}) + \nabla_{X_t^i} \log \frac{1}{N} \sum_{jx}^{N} \exp(-\alpha\, C(X_t^{i,jx}))$;
12         Update score
           $\phi^* \leftarrow \frac{1}{N_p} \sum_i^{N_p} [k^{\oplus}(X_t^i, X_t)\, \nabla_{X_t^i} \log p(X_t^i \mid \mathcal{O}) + \nabla_{X_t^i} k^{\oplus}(X_t^i, X_t)]$;
13     Update paths $X_t \leftarrow X_t + \epsilon\, \phi^*$;
14     Update prior $q(X_t \mid \mathcal{O}) \leftarrow p(X_t \mid \mathcal{O})\ t \leftarrow t + 1$;
```

5.4.1 Motion planning on 2-D terrain

Our first set of experiments consists of trajectory optimisation in a randomised 2-D terrain illustrated in Fig. 5.2. Regions of higher cost, or hills, are shown in a darker shade whereas valleys are in a lighter colour. The terrain is parameterised by a series of isotropic Multivariate Gaussian distributions placed randomly according to a Halton sequence and aggregated into a Gaussian Mixture Model denoted by p_{map}.

Paths are parameterised by natural cubic splines with $N_k = 2$ intermediary knots, apart from the start and goal state. Our goal is to find the best placement for these knots to find paths from origin to goal that avoid regions of high cost but are not too

	Cost	Steps
SigSVGD	**1056.0 (58.4)**	**189.3 (12.6)**
SVMPC	1396.4 (73.0)	239.1 (49.4)
MPPI	1740.7 (192.3)	290.8 (23.7)
CMA-ES†	—	—

Table 5.1: **Point-mass navigation results**. The table shows the mean and standard deviation for 20 episodes. *Cost* indicates the total accrued cost over the episode. CMA-ES cost is not shown as it couldn't complete the task on any episodes. *Steps* indicates the total amount of time-steps the controller needed to reach the goal. †CMA-ES couldn't complete any episodes, so results are omitted.

long. We adopt the following cost function in order to balance trajectory length and navigability:

$$C(\mathbf{x}_t) = \sum_{t\in[a,b]} \Big(p_{\text{map}}(\mathbf{x}_t) + 75\,\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2\Big), \tag{5.14}$$

where the ℓ^2-norm term is a piecewise linear approximation of the trajectory length. To ensure the approximation is valid each trajectory is decimated into 100 waypoints before being evaluated by Eq. (5.14).

The initial knots are randomly placed and the plots in Fig. 5.2 show the final 20 trajectories found with three different optimisation methods. Furthermore, the colour of each path depicts its normalised final cost. On the left we can see the solutions found with Batch Gradient Descent (BGD) and note how all trajectories converge to two modes of similar cost. The SVMP results are more diverse, but failed to capture one of the BGD modes. Also note how, when multiple trajectories converge to a single trough, the spline knots are pushed away by the repulsive force resulting in suboptimal solutions. On the other hand, the trajectories found by SigSVGD are not only more diverse, finding more homotopic solutions, but are also able to coexist in the narrow valleys. This is possible since the repulsive force is being computed in the signature space and not based on the placement of the knots. Furthermore, notice how for the same reason the paths are more direct and coordinated when compared to SVMP.

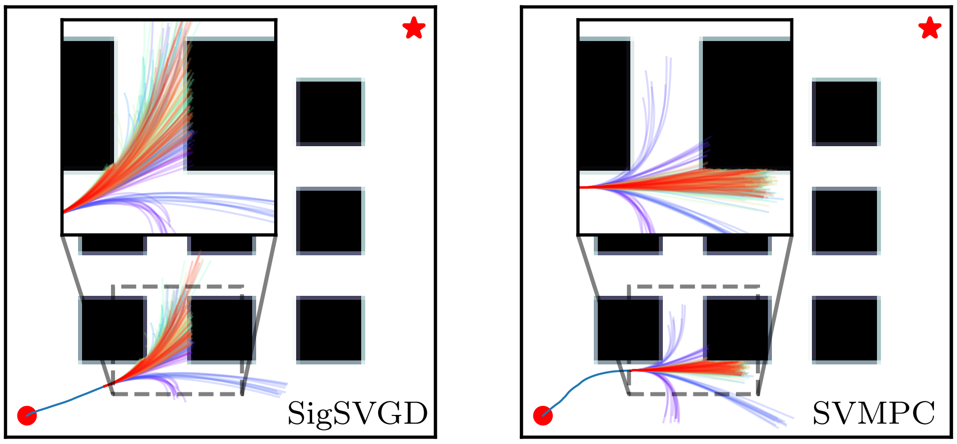


Figure 5.3: **Point-mass navigation trajectories**. The plot shows an intermediate time-step of the navigation task for SigSVGD, on the left, and SVMPC, on the right. An inset plot enlarges a patch of the map just ahead of the point-mass. The rollout colour indicate from which of the policies, i.e. paths in the optimisation, they originate, whereas fixed motion primitives are shown in purple. Note how rollouts generated by SigSVGD are more disperse, providing a better gradient for policy updates.

5.4.2 Point-mass navigation on an obstacle grid

Here, our goal is to demonstrate the benefits of applying the signature kernel MPC. To that end, we reproduce the point-mass planar navigation task presented in (Barcelos et al., 2021; Lambert et al., 2020) and compare SVMPC against and a modified implementation using SigSVGD. The objective is to navigate an holonomic point-mass robot from start to goal through an obstacle grid. Since the system dynamics is represented as a double integrator model with non-unitary mass m, the particle acceleration is given by $\ddot{\mathbf{x}} = m^{-1}\mathbf{u}$ and the control signal is the force applied to the point-mass. We adopt the same cost function as in (Barcelos et al., 2021), that is:

$$\begin{aligned} \mathcal{C}(\mathbf{x}_t, \mathbf{u}_t) &= 0.5\,\mathbf{e}_t^\mathsf{T}\mathbf{e}_t + 0.25\,\dot{\mathbf{x}}_t^{\mathsf{T}}\dot{\mathbf{x}}_t + 0.2\,\mathbf{u}_t^\mathsf{T}\mathbf{u}_t + \mathbb{1}\{\text{col.}\}\,p \\ \mathcal{C}_{\text{term}}(\mathbf{x}_t, \mathbf{u}_t) &= 1000\,\mathbf{e}_t^\mathsf{T}\mathbf{e}_t + 0.1\,\dot{\mathbf{x}}_t^{\mathsf{T}}\dot{\mathbf{x}}_t\,, \end{aligned}$$

where $\mathbf{e}_t = \mathbf{x}_t - \mathbf{x}_g$ is the instantaneous position error and $p = 10^6$ is the penalty when a collision happens.

To create a controlled environment with several multi-modal solutions, obstacles are placed equidistantly in a grid (see Fig. 5.3). The simulator performs a simple collision check based on the particle's state and prevents any future movement in case a collision is detected, simulating a crash. Barriers are also placed at the environment boundaries to prevent the robot from easily circumventing the obstacle grid. As the indicator function makes the cost function non-differentiable, we need to compute approximate gradients using Monte Carlo sampling (Lambert et al., 2020). Furthermore, since we are using a stochastic controller, we also include CMA-ES and MPPI (G. Williams et al., 2016) in the benchmark. A detailed account of the hyperparameters used in the experiment is presented in Appendix C.2.

In this experiment, each of the particles in the optimisation is a path that represents the mean of a stochastic control policy. Gradients for the policy updates are generated by sampling the control policies and evaluating *rollouts* via an implicit model of the environment. As CMA-ES only entertains a single solution at any given time, to make the results comparable we increase the amount of samples it evaluates at each step to be equivalent to the number of policies times the number of samples in SVMPC. One addition to the algorithm in (Lambert et al., 2020) is the inclusion of particles with predefined primitive control policies which are not optimised. For example, a policy which constantly applies the minimum, maximum, or no acceleration are all valid primitives. These primitive policies are also included in every candidate solution set of CMA-ES.

The inlay plot in Fig. 5.3 illustrates how SigSVGD promotes policies that are more diverse, covering more of the state-space on forward rollouts. The outcome can be seen on Table 5.1. SigSVGD finds lower cost policies and is able to reach the goal in fewer steps than SVMPC. Due to the dynamical nature of the problem, we are unable to run the optimisation for many iterations during each time-step, as we need to get actions from the controller at a fast rate. This poses a challenge to CMA-ES, which crashed on all episodes despite having a much larger number of samples per step.

5.4.3 Benchmark comparison on robotic manipulator

To test our approach on a more complex planning problem we compare batch gradient descent (i.e. parallel gradient descent on different initialisations), SVMP and SigSVGD in robotic manipulation problems generated using a benchmark released by Chamzas et al. (2022), MotionBenchMaker. A problem consists of a scene with ran-

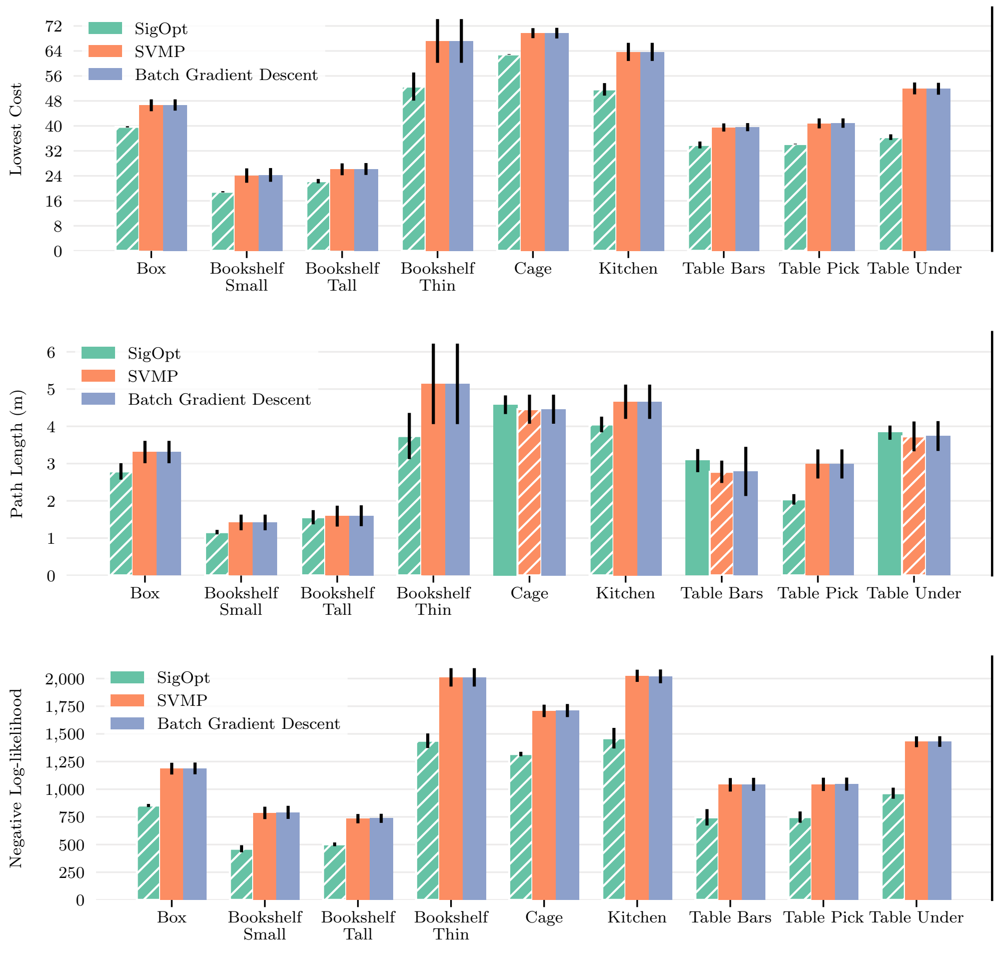
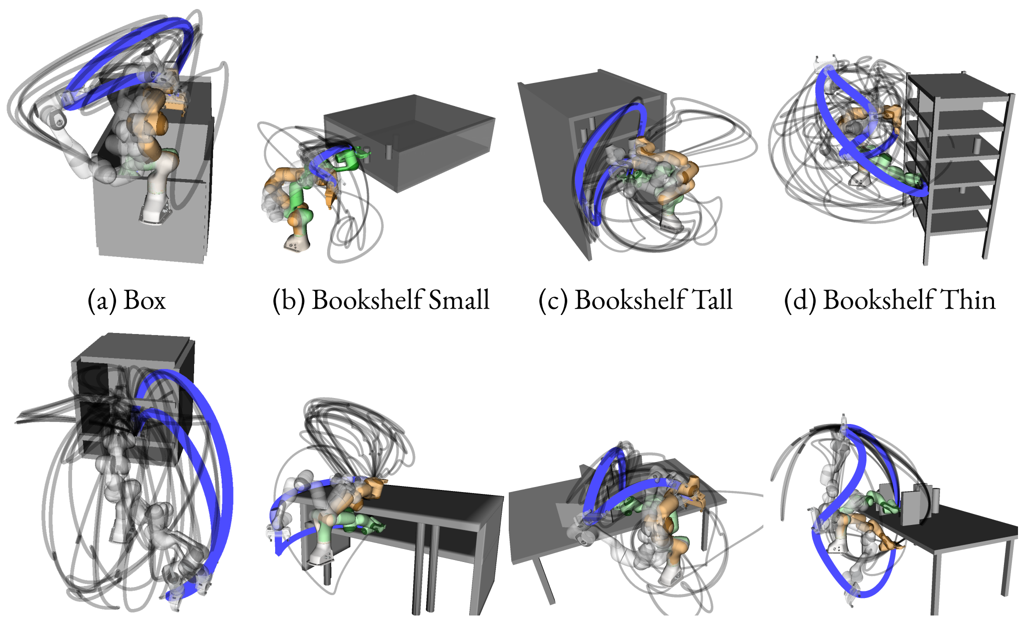


Figure 5.4: **Motion planning benchmark**. Results shown are the mean and standard deviation over 5 episodes for 4 distinct requests, totalling 20 iterations per scene. Best result is highlighted with a hatched bar. *Lowest cost* depicts the cost of the best trajectory found. *Path length* is the piecewise linear approximation of the end-effector trajectory length for the best trajectory. *NLL* indicates the negative log likelihood and, since we are using an exponential likelihood, represents the total cost of all sampled trajectories.



(a) Box (b) Bookshelf Small (c) Bookshelf Tall (d) Bookshelf Thin

(e) Cage (f) Table Bars (g) Table Pick (h) Table Under

Figure 5.5: **Visualisation of SigSVGD in the motion planning benchmark**. The *Blue* and *Grey* lines denote the end-effector's trajectories with the former highlighting the trajectory with the lowest cost. The *Orange* and *Green* tinted robot poses denote the start and target configuration, respectively. The translucent robot poses denote in-between configurations of the lowest-cost solution.

domly placed obstacles and a consistent request to move the manipulator from its starting pose to a target configuration. For each scene in the benchmark, we generate 4 different requests and run the optimisation with 5 random seeds for a total of 20 episodes per scene.

The robot used is a Franka Emika Panda with 7 DOF. The cost function is designed to generate trajectories that are smooth, collision-free and with a short displacement of the robot's end-effector. We once again resort to a fully-differentiable function to reduce the extraneous influence of approximating gradients with Monte Carlo samples. As is typical in motion planning, the optimisation is performed directly in *configuration space* (C-space), which simplifies the search for feasible plans. To reduce the sampling space and promote smooth trajectories, we once again parameterise the path of each of the robot joints with natural cubic splines, adopting 3 intermediary knots besides those at the initial and target poses.

Robot collision as continuous cost

Typically collision-checking is a binary check and non-differentiable. To generate differentiable collision checking with informative gradients, we resort to continuous occupancy grids. Occupancy grid maps are often generated from noisy and uncertain sensor measurement by discretising the space $\mathcal{W}$ where the robot operates (know as *workspace*) into grid-cells, where each cell represents an evenly spaced field of binary random variables that corresponds to the presence of an obstacle at the given location. However, the discontinuity in-between each cell means these grid maps are non-differentiable and not suitable for optimisation-based planning. A continuous analogue of an occupancy map can be generalised by a kernelised projection to high-dimensional spaces (Ramos & Ott, 2016) or with distance-based methods (Jones, Baerentzen & Sramek, 2006).

In this work we trade off the extra complexity of the methods previously mentioned for a coarser but simpler approach. Inspired by (Danielczuk et al., 2021), we learn the occupancy of each scene using a neural network as a universal function approximator. We train the network to approximate a continuous function that returns the likelihood of a robot configuration being occupied. The rationale for this choice is that, since all methods are optimised under the same conditions, the comparative results should not be substantially impacted by the overall quality of the map. Additionally, the trained network is fast to query and fast to obtain derivatives with respect to inputs, properties that are beneficial for querying of large batches of coordinates for motion planning.

Given a dataset of N pairs of coordinates and a binary value which indicates if the coordinate is occupied, i.e. $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{W} \subseteq \mathbb{R}^w$, and $y_i \in \{0, 1\}$, for $i = 1, \ldots, N$. The network then learns a mapping f_{col} between a coordinate of interest $\mathbf{x}$ and the probability of it being occupied, that is, $f_{\text{col}}(\mathbf{x}) = \mathbb{P}(y = 1 \mid \mathbf{x})$. A dataset of this format can be obtained, for instance, from depth sensors as point clouds. We model f_{col} as a fully-connected neural network, with tanh as the activation function between hidden layers, and $\mathsf{sigmoid}$ as the output layer. The final network is akin to a binary classification problem, which can be learned via a binary cross-entropy loss with gradient descent optimisers. As such, we can construct a collision cost function $f_{\text{col}} : \mathcal{W} \to \mathbb{R}$ that maps workspace coordinates into cost values associated at the corresponding locations.

A similar problem occurs when ascertaining whether a given configuration of the robot's joints is unfeasible, leading to a self-collision. We address this issue in a similar manner, by training a separate neural network to approximate a continuous function $f_{\text{s-col}}$ which maps configurations of the robot to the likelihood of they being in self-collision. More precisely, $f_{\text{s-col}} : \mathcal{Q} \to \mathbb{R}$, where $f_{\text{s-col}}(\mathbf{q}) = \mathbb{P}(y = 1 \mid \mathbf{q})$, for $\mathbf{q}_i \in \mathcal{Q} \subseteq \mathbb{R}^d$, and $y_i \in \{0, 1\}$. The dataset used to train $f_{\text{s-col}}$ is generated by randomly choosing configurations within the joint limits of the robot and performing a binary self-collision check provided by the robot's Application Programming Interface (API).

	SigSVGD		SVMP		Batch Gradient Descent	
Scene	Depth	Feasible Pct.	Depth	Feasible Pct.	Depth	Feasible Pct.
Box	3.74 (2.30)	**94.99 (3.78)**	**3.62 (1.95)**	94.96 (3.32)	3.63 (1.95)	94.97 (3.31)
Bookshelf Small	**1.32 (2.50)**	**96.63 (5.48)**	1.55 (2.19)	96.20 (4.68)	1.56 (2.20)	96.18 (4.71)
Bookshelf Tall	0.56 (1.78)	98.30 (4.65)	0.27 (0.60)	99.02 (1.76)	**0.27 (0.59)**	**99.03 (1.74)**
Bookshelf Thin	**2.78 (3.11)**	**94.59 (4.94)**	3.14 (3.50)	93.54 (5.57)	3.14 (3.50)	93.54 (5.57)
Cage	2.13 (1.82)	**96.12 (2.92)**	**2.00 (1.67)**	96.11 (2.89)	**2.00 (1.67)**	96.11 (2.89)
Kitchen	**9.82 (6.95)**	88.04 (9.85)	10.61 (6.45)	88.59 (6.21)	10.62 (6.71)	**88.61 (6.21)**
Table Bars	**9.46 (7.43)**	**92.42 (5.89)**	9.52 (8.05)	92.09 (6.69)	9.70 (8.44)	92.05 (6.85)
Table Pick	**0.22 (0.67)**	**99.56 (1.67)**	0.83 (1.04)	98.06 (2.62)	0.83 (1.02)	98.08 (2.43)
Table Under	**3.33 (2.60)**	**93.63 (5.36)**	5.16 (4.75)	90.19 (8.21)	5.18 (4.77)	90.06 (8.30)

Table 5.2: **Motion planning benchmark**. Results shown are the mean and standard deviation over 5 episodes for 4 distinct requests, totalling 20 iterations per scene. *Contact Depth* indicates the average collision depth of the trajectories found (in millimetres), if a collision happens. *Feasible Pct.* is the average percentage of the trajectory that is collision-free.

Bringing collision cost from workspace to configuration space

Collision checking requires information about the workspace geometry of the robot to determine whether it overlaps with objects in the environment. On the other hand, we assume that the robot movement is defined and optimised in C-space. The cost functions to shape robot behaviour are often defined in the Cartesian task space. We denote C-space as $\mathcal{Q} \subseteq \mathbb{R}^d$, where there are d joints in the case of a robotic manipulator. The joint configurations, $\mathbf{q} \in \mathcal{Q}$, are elements of the C-space, while Cartesian coordinates in task space are denoted as $\mathbf{x} \in \mathcal{W}$. We now outline the procedure of *pulling* a cost gradient defined in the workspace to the C-space.

We start by defining b body points on the robot, each with a forward kinematics function ψ_i mapping configurations to the Cartesian coordinates $\mathbf{x}_i$ at the body point, $\psi_i : \mathcal{Q} \to \mathcal{W}$, for each $i = 1, \dots, b$. Let the Jacobian of the forward kinematics functions w.r.t. the joint configurations be denoted as

$$\mathbf{J}(\cdot)^i_\psi = \frac{\mathrm{d}\psi_i}{\mathrm{d}\mathbf{q}}(\cdot). \tag{5.15}$$

The derivative of a cost potential $\mathcal{C}_{\text{col}}$ which operates on the body points, such as the occupancy cost potential, can then be *pulled* into the C-space with:

$$\nabla_\mathbf{q} \mathcal{C} = \sum_{i=1}^{b} \mathbf{J}(\mathbf{q})^i_\psi \, \nabla_\mathbf{x} \mathcal{C}, \tag{5.16}$$

which allows us to update trajectory in the C-space $\mathcal{Q}$ with cost in the Cartesian space $\mathcal{W}$.

Regularising path length and dynamical motions

Finally, the use of splines to interpolate the trajectories ensures smoothness in generated trajectories, but that does not necessarily imply in smooth dynamics for the manipulator. To visualise this, consider, for example, a trajectory in $\mathcal{Q}$ parameterised by a natural cubic spline. The configurations $\mathbf{q}$ in between each knot can be interpolated, resulting in a smooth trajectory of the robot end-effector in Euclidean coordinates in SE(3). However, the same end-effector trajectory could be traversed in a constant linear speed or with a jerky acceleration and deceleration motion. More specifically, if we use a fixed number of interpolated configurations between knots without care to impose dynamical restrictions to the simulator, knots that are further apart will result in motions with greater speed and acceleration since a larger distance would be covered during the same interval. To avoid these abrupt motions on the robots joints, we introduce the term $\mathcal{C}_{\text{dyn}}$ to the cost function, which penalises the linear distance between consecutive configurations:

$$\mathcal{C}_{\text{dyn}} = \sum_{i=2}^{p} \mathbf{w}^\mathsf{T} \| \mathbf{q}_i - \mathbf{q}_{i-1} \|_2, \tag{5.17}$$

where p is the number of intermediary configurations chosen when discretising the path spline and the weight $\mathbf{w}$ can be used to assign a higher importance to certain robot joints. We choose to adopt a vector $\mathbf{w}$ which is a linear interpolation from 1 to 0.7, where the higher value is assigned to the base joint of the manipulator and progressively reduced until the end-effector. A similar approach as the one presented in Eq. (5.17) can be used to penalise the length of the robot's trajectory in workspace. We include a final term to our cost function, C_{len}, that penalises exclusively the length of the end-effector path. This brings us to our final cost function:

$$C = 2.5\ C_{\text{len}} + 2.5\ C_{\text{dyn}} + C_{\text{col}} + 10\ C_{\text{s-col}}, \tag{5.18}$$

where each of the terms are respectively the cost for path length, path dynamics, collision with the environment and self-collision. The optimisation is carried out for 500 iterations and the kernel repulsive force is scheduled with cosine annealing (Loshchilov & Hutter, 2017). By reducing the repulsive force on the last portion of the optimisation, we allow trajectories at the same local minima to converge to the modes and are able to qualitatively measure the diversity of each approach.

The results shown on Fig. 5.4 demonstrate how SigSVGD achieves better results in almost all metrics for every scenario. The proper representation of paths results in better exploration of the configuration space and leads to better global properties of the solutions found. This can be seen in Fig. 5.5, which shows the end-effector paths for SigSVGD and SVMP. One of such paths is also illustrated in Fig. 5.1. Results found by SigSVGD also show a higher percentage of feasible trajectories and lower contact depths for rollouts in collision (see Table 5.2).

5.5 Summary

This work, to the best of our knowledge, is the first to introduce the use of path signatures for trajectory optimisation in robotics. We discuss how this transformation can be used as a canonical *linear* feature map to represent trajectories and how it possesses many desirable properties, such as invariance under time reparametrisation. We use these ideas to construct SigSVGD, a kernel method to solve control and motion planning problems in a variational inference setting. It approximates the posterior distribution over optimal paths with an empirical distribution comprised of a set of vector-valued particles which are all optimised in parallel.

In previous work it has been shown that approaching the optimisation from the variational perspective alleviates the problem of local optimality, providing a more diverse set of solutions. We argue that the use of signatures improves on previous work and can lead to even better global properties. Despite the signature poor scalability, we show how we can construct fast and paralellisable signature kernels by leveraging recent results in rough path theory. The RKHS induced by this kernel creates a structured space that captures the sequential nature of paths. This is demonstrated through an extensive set of experiments that the structure provided helps the functional optimisation, leading to better global solutions than equivalent methods without it. We hope the ideas herein presented will serve an inspiration for further research and stimulate a groundswell of new work capitalising on the benefits of signatures in many other fields within the robotics community.

CHAPTER 6

Conclusion and future work

In the course of this thesis, we have navigated a path from basic control problems to advanced approximate inference control solutions, exploring the intersections of model uncertainty, Bayesian inference, multi-modal inference, and variational methods. The aim was to develop a more robust and versatile framework for control, capable of handling a variety of challenges. The result is a series of methods: Double Likelihood-free Inference Stochastic Control (DISCO), which efficiently handles uncertainty propagation using the Unscented Transform; Dual Stein Variational Model Predictive Control (DuSt-MPC), which uses variational inference to jointly infer the control policy and model parameters; and Kernel Signature Variational Gradient Descent (SigSVGD), which takes advantage of the signature of path to improve particle diversity and find more diverse solutions. Each of these has been carefully crafted to answer a unique set of challenges in robotics, from model uncertainty and the reality gap to the efficient approximation of the posterior distribution over control policies and model parameters. This chapter summarises the thesis's contributions and discusses possible future research directions.

6.1 Contributions

6.1.1 Incorporating model uncertainty in control as inference

Chapter 3 introduced the concept of incorporating model uncertainty and advanced Bayesian inference methods into stochastic model-based control. This approach introduces the foundation for an adaptive controller framework, capable of effectively dealing with the reality gap and covariate shift. By formally incorporating uncertainty over parameters into an SNMPC controller, we found that this led to a more accurate assessment of the environment and improved performance. To efficiently evaluate the cost of future paths the unscented transform was introduced as a means to propagate this uncertainty, reducing the computational burden on trajectory samples and paving the way for more efficient control.

6.1.2 Joint estimation of model and policy parameters

However, we noticed that there is a trade-off between the efficiency gained in sampling and accuracy when using moment matching. This motivated further exploration in Chapter 4 where we presented DuSt-MPC, a method to perform multi-modal inference under model uncertainty. DuSt-MPC encapsulates gathered observations into a prior distribution over model parameters and refines such distribution in real time, adapting the control policy to changes in the environment. Another major contribution of DuSt-MPC is that it can be applied with non-differentiable black-box simulators, since the gradients can be approximated with Monte Carlo samples. One such application is to reduce the reality gap in complex sim-to-real applications, as discussed by Possas et al. (2020). Despite the great strides made by DuSt-MPC, we identified limitations related to the lack of diversity once the dimensionality of the inference problem is large (Zhuo et al., 2018).

6.1.3 Addressing high-dimensional problems

In response to these challenges, Chapter 5 presented the use of path signatures for trajectory optimisation in robotics. The introduction of signatures, as implemented in SigSVGD, presents a novel way to represent trajectories and compute their similarities in a projected feature space. This approach leverages the desirable properties of trajectories, promotes higher particle diversity on the approximated posterior distribution, and thus leads to optimisation solutions with better global properties. Furthermore, SigSVGD also introduces a novel and statistically sound manner of handling input constraints with the use of hyper-priors over paths. Despite the poor scalability of the signature method, we found that it could still be used effectively through recent results in rough path theory (Salvi et al., 2021).

The work presented in this thesis is one of the first to combine these various elements into a cohesive whole, pushing the boundaries of what is possible in planning and control policy optimisation under model uncertainty. It is clear from our experiments that the integration of model uncertainty and variational inference can significantly improve the performance of control systems, particularly in situations where disturbances or unmodelled phenomena are present.

Each step, from incorporating model uncertainty and Bayesian inference into SN-MPC, developing methods for multi-modal inference under model uncertainty, to introducing the use of path signatures for trajectory optimisation, has led to improved performance and greater resilience in the face of challenges such as the reality gap, covariate shift, and parametric uncertainty. The experiments have provided strong evidence of these improvements, demonstrating that our methods outperform existing approaches. In conclusion, we hope this thesis constitutes a significant contribution to the field of robotics and control systems and that it paves the way for future research on how to best apply these ideas in varied applications.

6.2 Future work

However, it is important to note that there are still many challenges to overcome and questions to be answered. The trade-off between efficiency and accuracy when using moment matching, the problem of diversity in other high-dimensionality settings, and the scalability issues of the signature method are all areas that warrant further investigation. Looking forward, it is our hope that this work will stimulate further research into these and other related areas. Here we list but a few potential avenues to explore.

6.2.1 Exploring new uses for the path signature in robotics

The use of path signatures in planning and control problems is perhaps the most palpable application in robotics, but it is by no means the only one. Like we discussed in Chapter 5, the signature is a canonical feature map for any kind of sequential data (Boedihardjo et al., 2016). As such, all kinds of time-series data are candidates sources for a signature transform. Evaluating if such transformation would be beneficial, for example, in policies trained on end-to-end sequential images (Levine et al., 2015) or perhaps in recursive Bayesian estimation in filtering applications (Castellano-Quero, Fernández-Madrigal & García-Cerezo, 2020) is a promising pathway.

For example, exploring pathways to efficiently retrain estimates online for practical experiments with time-variant parameters could be a crucial step towards the generalisation of control policies for autonomous robots operating under varying environments and configurations. Similarly, the use of black-box simulators, such as data-driven function approximators, could further enhance the versatility of the methods we have presented.

6.2.2 Conditioning policies on states

Another interesting research pathway is to explore how to persistently condition the learned policy on the current state of the system. The methods presented are all *stateless*, that is, a solution is recalculated every time for each situation the agent is experiencing presently. Additionally, it would be beneficial if such policy could be updated online, with the gathering of new observations from the real system. In other words, unlike the distribution over the model parameters which is constantly refined, the policy does not improve from experience. We anticipate that extensive datasets would be needed to either supervised or semi-supervised train such a policy, but the advances in foundational models in computer vision could facilitate the creation of simulated or augmented datasets in that direction. For example, models capable of generating text-to-image are already a reality (Ramesh et al., 2021). These could in turn be used to generate 3-D scenes (Mildenhall et al., 2020) and converting such scenes to assets of a simulator is the missing link.

6.2.3 Decision theory with multi-modal posteriors

We have presented methods which are capable of approximating multi-modal posterior distributions, but not much focus was given on how to best leverage this information on mission critical applications. The most straightforward solution is to pick the solution pertaining to the most likely mode, however this does not take risk into consideration. Another approach is to sample the posterior for solutions, e.g. actions or paths, however this may lead to inconsistent outcomes when several modes are likely. For example, when changing lanes in traffic weaving, one sample may be swerving in one direction and the subsequent sample might go in the opposite way (Schmerling et al., 2018). The scenario described is studied in *sequential decision making*, and often solved using imitation learning, like Behavioral Cloning (BC), or RL. One possible avenue of research is to learn a latent space representation of previous actions that can be used to condition the posterior distribution, making sampling for a solution more amenable. Another possibility is to evaluate utility functions that could encode the desired behaviour, e.g. assign a higher likelihood to safer solutions, and hence reduce the problem multi-modality.

APPENDIX A

Supplementary material for Chapter 3

A.1 Further details on performed experiments

The hyper-parameters used in the experimental section of Chapter 3 are listed next, on Table A.1 and Table A.2. For both experiments, the unscented transform secondary scaling ($\varkappa$) and minimum control ($\tilde{\mathbf{u}}$) were set to zero. Note that, as the random seeds were not controlled, slight variations are expected when reproducing the results. Similarly, the update of the posterior distribution approximation, $q(\boldsymbol{\xi} \mid \mathcal{D})$, will depend on $\mathcal{D}$ and therefore will vary in every execution.

Table A.1: Hyper-parameters for the inverted pendulum experiment in Chapter 3.

Parameter	**Value**
Sampled actions, $N_{\mathbf{u}}$	500
Control horizon, H	30
Inverse temperature, α	10
Control authority, $\mathbf{\Sigma}$	1
Minimum control, $\tilde{\mathbf{u}}$	0
Instant state cost, $\mathcal{C}_{\text{inst}}$	$50\cos(\theta - 1)^2 + \dot{\theta}^2$
Terminal state cost, $\mathcal{C}_{\text{term}}$	0
Sigma points, L	5
UT Secondary scaling (κ)	0
UT Spread, α	0.5
UT scalar, o	2
Prior distribution, $p(\boldsymbol{\xi})$	
- over pole length l	$\mathcal{U}(0.1, 5)$
- over pole mass m	$\mathcal{U}(0.1, 5)$
Posterior distribution, $q_{\phi}(\boldsymbol{\xi} \mid \mathcal{D})$	
- over pole length l	$\mathcal{N}(0.89, 0.01)$
- over pole mass m	$\mathcal{N}(0.9, 0.03)$

Table A.2: Hyper-parameters for the skid-steer experiment in Chapter 3.

Parameter	**Value**
Sampled actions, $N_{\mathbf{u}}$	400
Control horizon, H	50
Inverse temperature, α	0.1
Control authority, $\mathbf{\Sigma}$	0.25
Minimum control, $\tilde{\mathbf{u}}$	0
Instant state cost, $\mathcal{C}_{\text{inst}}$	$\sqrt{d_t^2 + (\dot{\mathbf{x}}_t - \dot{\mathbf{x}}_0)^2}$
Terminal state cost, $\mathcal{C}_{\text{term}}$	0

Continued on next page

Table A.2: (continued)

Parameter	**Value**
Sigma points, L	7
UT Secondary scaling, κ	0
UT Spread, α	0.5
UT scalar, o	2
Prior distribution, $p(\boldsymbol{\xi})$	
- over x_{ICR}	$\mathcal{U}(0, 0.5)$
- over r_{w}	$\mathcal{U}(0, 0.5)$
- over a_{w}	$\mathcal{U}(0.1, 0.5)$
Posterior distribution, $q_{\phi}(\boldsymbol{\xi} \mid \mathcal{D})$	
- $[x_{\text{ICR}}, r_{\text{w}}, a_{\text{w}}]^{\mathsf{T}}$	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- $\boldsymbol{\mu}$	$\begin{bmatrix} 0.238 & 0.061 & 0.415 \end{bmatrix}^{\mathsf{T}}$
- $\boldsymbol{\Sigma} \times 10^{-3}$	$\begin{bmatrix} 0.13 & -0.03 & -0.04 \\ -0.03 & 0.15 & 0.03 \\ -0.04 & 0.03 & 0.09 \end{bmatrix}$

APPENDIX B

Supplementary material for Chapter 4

B.1 Further details on performed experiments

The hyperparameters used in the experimental section of Chapter 4 are listed next, on Tables B.1 to B.3. For the inverted pendulum experiment, the unscented transform secondary scaling ($\varkappa$) and minimum control ($\tilde{\mathbf{u}}$) were set to zero. Note that, as the random seeds were not controlled, slight variations are expected when reproducing the results. Similarly, the update of the posterior distribution approximation, $q(\boldsymbol{\xi} \mid \mathcal{D})$, will depend on $\mathcal{D}$ and therefore will vary in every execution.

Table B.1: Hyper-parameters for the inverted pendulum experiment in Chapter 4.

Parameter	**Inverted Pendulum**
Initial state, $\mathbf{x}_0$	[3 rad, 0 rad/sec]
Environment maximum velocity	8 rad/sec
Environment maximum acceleration/torque	2 N m
Policy samples, N_a	32
Dynamics samples, N_s	8
Cost Likelihood inverse temperature, α	1.0
Control authority, $\mathbf{\Sigma}$	2.0^2
Control horizon, H	20
Number of policies, N_π	3
Policy Kernel, $k_\pi(\cdot,\cdot)$	Radial Basis Function
Policy Kernel bandwidth selection	Silverman's rule
Policy prior covariance, $\mathbf{\Sigma}_\mathrm{a}$	2.0^2
Policy step size, ϵ	2.0
Dynamics prior distribution	
- mass, m	$\mathcal{U}(0.6, 1.3)$
- length, l	$\mathcal{U}(0.6, 1.3)$
Dynamics particles, N_ξ	50
Dynamics Kernel, $k_\xi(\cdot,\cdot)$	Radial Basis Function
Dynamics GMM covariance, $\mathbf{\Sigma}_\mathrm{s}$	Improved Sheather Jones
Dynamics likelihood covariance, $\mathbf{\Sigma}_\mathrm{obs}$	0.1^2
Dynamics update steps, L	20
Dynamics step size, ϵ	1×10^{-3}
Dynamics in log space	No
Unscented Transform spread, α	0.5

Table B.2: Hyper-parameters for the navigation experiment in Chapter 4.

Parameter	Point-mass Navigation
Policy samples, N_a	64
Dynamics samples, N_s	4
Cost Likelihood inverse temperature, α	1.0
Control authority, $\mathbf{\Sigma}$	5.0^2
Control horizon, H	40
Number of policies, N_π	6
Policy Kernel, $k_\pi(\cdot,\cdot)$	Radial Basis Function
Policy Kernel bandwidth selection	Silverman's rule
Policy prior covariance, $\mathbf{\Sigma}_\text{a}$	5.0^2
Policy step size, ϵ	100.0
Dynamics prior distribution	$\mathcal{N}(2, 0.1^2)$
Dynamics particles, N_ξ	50
Dynamics Kernel, $k_\xi(\cdot,\cdot)$	Radial Basis Function
Dynamics GMM covariance, $\mathbf{\Sigma}_\text{s}$	0.25^2
Dynamics likelihood covariance, $\mathbf{\Sigma}_\text{obs}$	0.1^2
Dynamics update steps, L	20
Dynamics step size, ϵ	1×10^{-2}
Dynamics in log space	Yes

Table B.3: Hyper-parameters for the trajectory tracking experiment in Chapter 4.

Parameter	Traj. Tracking
Policy samples, N_a	50
Dynamics samples, N_s	4
Cost Likelihood inverse temperature, α	1.0
Control authority, $\mathbf{\Sigma}$	0.1^2
Control horizon, H	20
Number of policies, N_π	2

Continued on next page

Table B.3: (continued)

Parameter	**Traj. Tracking**
Policy Kernel, $k_\pi(\cdot,\cdot)$	Radial Basis Function
Policy Kernel bandwidth selection	Silverman's rule
Policy prior covariance, $\mathbf{\Sigma}_\mathrm{a}$	1.0^2
Policy step size, ϵ	2×10^{-2}
Dynamics prior distribution	$\mathcal{N}(0.5, 0.2^2)$
Dynamics particles, N_ξ	50
Dynamics Kernel, $k_\xi(\cdot,\cdot)$	Radial Basis Function
Dynamics GMM covariance, $\mathbf{\Sigma}_\mathrm{s}$	0.0625^2
Dynamics likelihood covariance, $\mathbf{\Sigma}_\mathrm{obs}$	0.1^2
Dynamics update steps, L	5
Dynamics step size, ϵ	5×10^{-2}
Dynamics in log space	No

B.2 Considerations on action selection

Once policy parameters have been updated according to Eq. (4.11), we can update the policy for the current step, π_{θ_t}. However, defining the updated policy is not enough, as we still need to determine which immediate control action should be sent to the system. There are many options which could be considered at this stage. One alternative would be to take the expected value at each time-step, although that could compromise the multi-modality of the solution found. Other options would be to consider the modes π_{θ_t} of at each horizon step or sample the actions directly. Finally, we adopt the choice of computing the probabilistic weight of each particle and choosing the highest weighted particle as the action sequence for the current step. More formally:

$$\omega_i = \frac{p(\mathcal{O} \mid \theta_t^i, \xi)\, q(\theta_{t-1}^i)}{\sum_{j=1}^m p(\mathcal{O} \mid \theta_t^j, \xi)\, q(\theta_{t-1}^j)} \approx p(\theta_t^i \mid \mathcal{O}, \xi). \tag{B.1}$$

And finally, the action deployed to the system would be given by $U_t^{n^*} = \theta_t^{n^*}$, where $n^* = \arg\max_n \omega_t^n$.

B.3 Considerations on covariate shift

The proposed inference method assumes that the parameters of the dynamical model are fixed over time. Note that, even if the distribution over parameters changes over time, this remains a plausible consideration, given that the control loop will likely have a relatively high frequency when compared to the environment covariate shift. This also ensures that trajectories generated in simulation are consistent and that the resulting changes are being governed by the variations in the control actions and not the environment.

However, it is also clear that the method is intrinsically adaptable to changes in the environment, as long as there is a minimum probability of the latent parameter being feasible under the prior distribution $q(\xi \mid \mathcal{D}_t)$. Too see this, consider the case where there is an abrupt change in the environment (e.g. the agent picks-up some load or the type of terrain changes). In this situation, $q(\xi \mid \mathcal{D}_{t-1})$ would behave as if a poorly specified prior, meaning that as long the probability density around the true distribution $p(\xi)$ is non-zero, we would still converge to the true distribution, albeit requiring further gradient steps.

In practice, the more data we gather to corroborate a given parameter set, the more concentrated the distribution would be around a given location in the parameter space and the longer it would take to transport the probability mass to other regions of the parameter space. This could be controlled by including heuristic weight terms to the likelihood and prior in Eq. (4.14). However, we deliberately choose not to include extra hyper-parameters based on the hypothesis that the control loop is significantly faster than the changes in the environment, which in general allows the system to gather a few dozens or possibly more observations before converging to a good estimate of ξ.

B.4 Bias on joint inference of policy and dynamics

Note that, if we take the gradient of Eq. (4.8) w.r.t. ξ, we get:

$$\begin{aligned}\nabla_\xi \, p(\theta_t, \xi \mid \mathcal{O}, \mathcal{D}) &= \nabla_\xi [p(\theta_t \mid \mathcal{O}, \xi) \, p(\xi \mid \mathcal{D})] \\ &= p(\xi \mid \mathcal{D}) \, \nabla_\xi \, p(\theta_t \mid \mathcal{O}, \xi) + p(\theta_t \mid \mathcal{O}, \xi) \, \nabla_\xi \, p(\xi \mid \mathcal{D}).\end{aligned} \tag{B.2}$$

The first term on the right-hand side (RHS) of the equation above indicates that the optimality gradient of *simulated* trajectories would contribute to the update of $\boldsymbol{\xi}$. This implies that the estimation of $p(\boldsymbol{\xi})$ would be driven to regions that would yield lower cost policies, i.e. possibly due to easier to control dynamics. Naturally, it is important that the likelihood of $\boldsymbol{\xi}$ be taken into account during the policy updates, but we don't want the inference of the *physical* parameters to be biased by our optimality measure. In other words, the distribution over $\boldsymbol{\xi}$ shouldn't conform to whatever would benefit the optimality policy, but the other way around.

APPENDIX C

Supplementary material for Chapter 5

C.1 Path following example

As a motivating example in Fig. C.1 we depict the results of a simple two-dimensional path following task. The goal is to reduce the error between the desired path and candidate paths. Since we want the error to be as small as possible, the optimal path is one centred at the origin across time. The objective function is defined as a correlated multivariate Normal distribution across 10 consecutive discrete time-steps such that the optimality likelihood is computed for the entire discretised path. As the cost function is convex and we are optimising the paths directly—i.e. not searching for an indirect policy that generates the candidate paths—the solution is trivial. Nonetheless, the example is useful to illustrate the differences between SigSVGD and SVMP.

The initial paths are sampled from a uniform distribution and optimised with SVMP and SigSVGD for 200 iterations. The length scale of the squared-exponential kernel is computed according to Silverman's rule (Silverman, 1986) based on the initial sample for SigSVGD and updated at each iteration using the same method for SVMP. The results in Fig. C.1 show how both methods are able to promote diversity on the resulting paths. However, close inspection of the SVMP solution illustrates how coordinates of the candidate paths at each time-step are optimised without coordination, resulting in many paths crisscrossing and non-optimal paths close to the origin. Conversely, SigSVGD is promoting diversity of complete paths, rather than coordinates at each cross-sectional time-step, resulting in more direct paths with higher optimality likelihood.

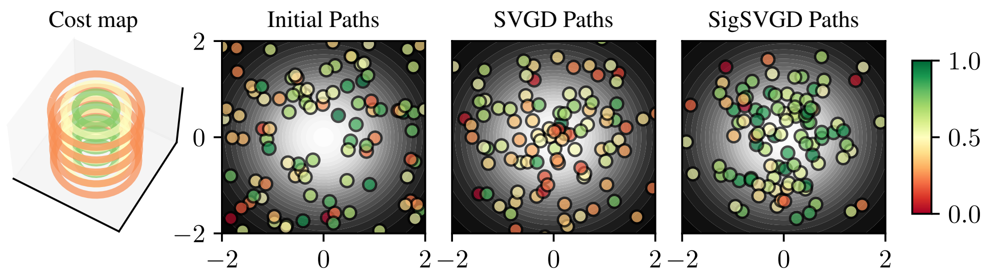


Figure C.1: **Qualitative analysis of trajectory tracking task**. *Left*: Contour plot of the optimality distribution over sequential time-steps (on z-axis). *Centre-left*: Cross-section plot at a given time-step of initial path coordinates. The colour of each path indicates its normalised optimality probability. *Centre-right*: Cross-section plot of the paths after SVGD optimisation. The sampled paths are diverse and capture the variance of the target distribution. Note, however, that many non-optimal trajectories are close to the origin due to the lack of coordination between consecutive time-steps. *Right*: Cross-section plot of the paths after SigSVGD optimisation. Note how we achieve both diversity and a concentration of optimal paths near the origin.

C.2 Experiments hyper-parameters

In Tables C.1 to C.3 we present the relevant hyper-parameters to reproduce the results in the paper. It is worth mentioning that the terrain in the 2-D motion planning is randomly generated and will vary on each simulation. Another source of randomness arises when using Monte Carlo samples to approximate the gradient of the log posterior distribution. Furthermore, due to the stochastic nature of the initial placement of the spline knots, results will vary despite using analytic gradients.

Table C.1: Hyper-parameters for the 2-D terrain experiment.

Parameter	**2-D Terrain**
Initial state, $\mathbf{x}_s$	$[0.25, 0.75]$
Number of spline knots, N_k	4
Number of particles, N_p	20
Particle prior	Uniform
Cost likelihood inverse temperature, λ	1.0
Stationary kernel, $k(\cdot,\cdot)$	Squared-exponential
Stationary Kernel bandwidth, σ	1.5
Signature kernel bandwidth, σ	1.5
Signature kernel degree, d	4
Optimiser class	Adam
Learning rate, ϵ	5×10^{-2}

Table C.2: Hyper-parameters for the navigation experiment.

Parameter	**Point-mass Navigation**
Initial state, $\mathbf{x}_s$	$[-1.8, -1.8]$
Environment maximum velocity	5 m/sec
Number of particles, N_p	30
Particle prior	$\mathcal{N}(X, \mathbf{1})$
Number of action samples, N_a	10
Cost likelihood inverse temperature, λ	1.0
Control authority, $\mathbf{\Sigma}$	25
Control horizon, H	30

Continued on next page

Table C.2: (continued)

Parameter	Point-mass Navigation
Stationary kernel, $k(\cdot,\cdot)$	Squared-exponential
Stationary Kernel bandwidth, σ	Silverman's rule
Signature kernel bandwidth, σ	5.65
Signature kernel degree, d	3
Optimiser class	Adam
Learning rate, ϵ	1

Table C.3: Hyper-parameters for the manipulator benchmark experiment.

Parameter	Manipulator Benchmark
Initial state, $\mathbf{x}_s$	Problem dependent
Number of spline knots, N_k	5
Number of particles, N_p	20
Particle prior	Uniform
Cost likelihood inverse temperature, λ	1.0
Stationary kernel, $k(\cdot,\cdot)$	Squared-exponential
Stationary Kernel bandwidth, σ	1.5
Signature kernel bandwidth, σ	1.5
Signature kernel degree, d	6
Optimiser class	Adam
Learning rate, ϵ	1×10^{-3}

C.3 Including hyper-priors in SigSVGD

As mentioned in Section 5.3.2, if one wants to constraint the feasible set of the SVGD optimisation a *hyper-prior* can be included in the algorithm. Let $h(\cdot)$ be a hyper-prior and $p(\cdot)$ the prior distribution over particles $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and recall that the *score function* at each update is computed according to

$$\boldsymbol{\phi}^*(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim p}\big[k^{\oplus}(\mathbf{y},\mathbf{x})\, \nabla_{\mathbf{y}} \log p(\mathbf{y} \mid \mathcal{O}) + \nabla_{\mathbf{y}} k^{\oplus}(\mathbf{y},\mathbf{x})\big],$$

where the posterior distribution can be factored in $\log p(\mathbf{y} \mid \mathcal{O}) = \mathcal{L}(\mathcal{O} \mid \mathbf{y}) + \log p(\mathbf{y})$. We can include the hyper-prior in the formulation by variable substitution. Let $\log \hat{p}(\cdot) = \log p(\cdot) + \log h(\cdot)$, then

$$\begin{aligned}
\boldsymbol{\phi}^*(\mathbf{x}) &= \mathbb{E}_{\mathbf{y}\sim p}\big[k^{\oplus}(\mathbf{y},\mathbf{x})\,\nabla_{\mathbf{y}} \log p(\mathbf{y} \mid \mathcal{O}) + \nabla_{\mathbf{x}} k^{\oplus}(\mathbf{y},\mathbf{x})\big] \\
\boldsymbol{\phi}^*(\mathbf{x}) &= \mathbb{E}_{\mathbf{y}\sim p}\Big[k^{\oplus}(\mathbf{y},\mathbf{x})\,\nabla_{\mathbf{y}}\big[\mathcal{L}(\mathcal{O} \mid \mathbf{y}) + \log \hat{p}(\mathbf{y})\big] + \nabla_{\mathbf{y}} k^{\oplus}(\mathbf{y},\mathbf{x})\Big] \\
\boldsymbol{\phi}^*(\mathbf{x}) &= \mathbb{E}_{\mathbf{y}\sim p}\Big[k^{\oplus}(\mathbf{y},\mathbf{x})\,\nabla_{\mathbf{y}}\big[\mathcal{L}(\mathcal{O} \mid \mathbf{y}) + \log p(\mathbf{y}) + \log h(\mathbf{y})\big] + \nabla_{\mathbf{y}} k^{\oplus}(\mathbf{y},\mathbf{x})\Big],
\end{aligned}$$

where $h(\cdot)$ can be any differentiable probability density function. ■

Bibliography

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous Helicopter Aerobatics through Apprenticeship Learning. *The International Journal of Robotics Research*, *29*(13), 1608–1639. https://doi.org/10.1177/0278364910371999

Abraham, I., Handa, A., Ratliff, N., Lowrey, K., Murphey, T. D., & Fox, D. (2020). Model-Based Generalization Under Parameter Uncertainty Using Path Integral Control. *IEEE Robotics and Automation Letters*, *5*(2), 2864–2871. https://doi.org/10.1109/LRA.2020.2972836

Agarwal, P., Kumar, S., Ryde, J., Corso, J., Krovi, V., Ahmed, N., Schoenberg, J., Campbell, M., Bloesch, M., Hutter, M., Hoepflinger, M., Leutenegger, S., Gehring, C., Remy, C. D., Siegwart, R., Brookshire, J., Teller, S., Bryson, M., Johnson-Roberson, M., ... Giordano, P. R. (2013). On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference. In *Robotics: Science and Systems VIII* (pp. 353–360). MIT Press. Retrieved February 20, 2023, from https://ieeexplore.ieee.org/document/6577934

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Al-Bluwi, I., Siméon, T., & Cortés, J. (2012). Motion planning algorithms for molecular simulations: A survey. *Computer Science Review*, *6*(4), 125–143. https://doi.org/10.1016/j.cosrev.2012.07.002

Améndola, C., Friz, P., & Sturmfels, B. (2019). Varieties of Signature Tensors. *Forum of Mathematics, Sigma*, *7*, e10. https://doi.org/10.1017/fms.2019.3

Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*. https://doi.org/10.1023/A:1020281327116

Andrychowicz, O. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., & Zaremba, W. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, *39*(1), 3–20. https://doi.org/10.1177/0278364919887447

Arruda, E., Mathew, M. J., Kopicki, M., Mistry, M., Azad, M., & Wyatt, J. L. (2017). Uncertainty averse pushing with model predictive path integral control. *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 497–502. https://doi.org/10.1109/HUMANOIDS.2017.8246918

Atkeson, C. G., & Santamaria, J. C. (1997). A comparison of direct and model-based reinforcement learning. *Proceedings of International Conference on Robotics and Automation*, *4*, 3557–3564 vol.4. https://doi.org/10.1109/ROBOT.1997.606886

Barcelos, L., & Camponogara, E. (2010). Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, *18*(1). https://doi.org/10.1016/j.trc.2009.04.022

Barcelos, L., Lambert, A., Oliveira, R., Borges, P., Boots, B., & Ramos, F. (2021, July 12–16). Dual online stein variational inference for control and dynamics. In D. A. Shell, M. Toussaint & M. A. Hsieh (Eds.), *Robotics: Science and systems XVII (RSS)* (pp. 1–12). https://doi.org/10.15607/RSS.2021.XVII.068

Barcelos, L., Oliveira, R., Possas, R., Ott, L., & Ramos, F. (2020). DISCO: Double Likelihood-Free Inference Stochastic Control. *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 7. https://doi.org/978-1-7281-7395-5/20

Barfoot, T., Hay Tong, C., & Sarkka, S. (2014). Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression. *Robotics: Science and Systems X*, 1–9. https://doi.org/10.15607/RSS.2014.X.001

Beaumont, M. A., Zhang, W., & Balding, D. J. (2002). Approximate Bayesian Computation in Population Genetics. *Genetics*, *162*(4), 2025–2035. https://doi.org/10.1093/genetics/162.4.2025

Berntorp, K., Olofsson, B., Lundahl, K., & Nielsen, L. (2014). Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres. *Vehicle System Dynamics*, *52*(10), 1304–1332. https://doi.org/10.1080/00423114.2014.939094

Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control, Vol. 2* (Fourth edition, Vol. 2). Athena Scientific.
OCLC: 930844423.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877. https://doi.org/10.1080/01621459.2017.1285773

Boedihardjo, H., Geng, X., Lyons, T., & Yang, D. (2016). The signature of a rough path: Uniqueness. *Advances in Mathematics*, *293*, 720–737. https://doi.org/10.1016/j.aim.2016.02.011

Bonassi, F. V., & West, M. (2015). Sequential Monte Carlo with Adaptive Weights for Approximate Bayesian Computation. *Bayesian Analysis*, *10*(1), 171–187. https://doi.org/10.1214/14-BA891

Botev, Z. I., Grotowski, J. F., & Kroese, D. P. (2010). Kernel density estimation via diffusion. *The Annals of Statistics*, *38*(5), 2916–2957. https://doi.org/10.1214/10-AOS799

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., & L'Ecuyer, P. (2013). The Cross-Entropy Method for Optimization. In *Handbook of Statistics* (pp. 35–59, Vol. 31). Elsevier. https://doi.org/10.1016/B978-0-444-53859-8.00003-5

Bradford, E., & Imsland, L. (2018). Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. *IFAC-PapersOnLine*, *51*(18), 417–422. https://doi.org/10.1016/j.ifacol.2018.09.336

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016, June 5). *OpenAI Gym*. Retrieved May 2, 2019, from http://arxiv.org/abs/1606.01540

Buchanan, R., Bandyopadhyay, T., Bjelonic, M., Wellhausen, L., Hutter, M., & Kottege, N. (2019). Walking Posture Adaptation for Legged Robot Navigation in Confined Spaces. *IEEE Robotics and Automation Letters*, *4*(2), 2148–2155. https://doi.org/10.1109/LRA.2019.2899664

Burden, A. G., Caldwell, G. A., & Guertler, M. R. (2022). Towards human–robot collaboration in construction: Current cobot trends and forecasts. *Constr Robot*, *6*(3), 209–220. https://doi.org/10.1007/s41693-022-00085-0

Byravan, A., Boots, B., Srinivasa, S. S., & Fox, D. (2014). Space-time functional gradient optimization for motion planning. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 6499–6506. https://doi.org/10.1109/ICRA.2014.6907818

Cai, K., Wang, C., Song, S., Chen, H., & Meng, M. Q.-H. (2021). Risk-Aware Path Planning Under Uncertainty in Dynamic Environments. *J Intell Robot Syst*, *101*(3), 47. https://doi.org/10.1007/s10846-021-01323-3

Camacho, E. F., & Alba, C. B. (2013). *Model Predictive Control* (2nd ed.). Springer-Verlag. https://link.springer.com/book/10.1007/978-0-85729-398-5

Castellano-Quero, M., Fernández-Madrigal, J.-A., & García-Cerezo, A.-J. (2020). Statistical Study of the Performance of Recursive Bayesian Filters with Abnormal Observations from Range Sensors. *Sensors*, *20*(15), 4159. https://doi.org/10.3390/s20154159

Chamzas, C., Quintero-Pena, C., Kingston, Z., Orthey, A., Rakita, D., Gleicher, M., Toussaint, M., & Kavraki, L. E. (2022). MotionBenchMaker: A Tool to Generate and Benchmark Motion Planning Datasets. *IEEE Robot. Autom. Lett.*, *7*(2), 882–889. https://doi.org/10.1109/LRA.2021.3133603

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., & Fox, D. (2018, October 12). *Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience*. Retrieved May 17, 2019, from http://arxiv.org/abs/1810.05687

Chen, K.-T. (1954). Iterated Integrals and Exponential Homomorphisms. *Proceedings of the London Mathematical Society*, *s3-4*(1), 502–512. https://doi.org/10.1112/plms/s3-4.1.502

Chen, K.-T. (1958). Integration of Paths-A Faithful Representation of Paths by Noncommutative Formal Power Series. *Transactions of the American Mathematical Society*, *89*(2), 395–407. https://doi.org/10.2307/1993193

Chen, K.-T. (1977). Iterated path integrals. *Bulletin of the American Mathematical Society*, *83*, 831–879.

Chevyrev, I., & Kormilitzin, A. (2016, March 11). *A Primer on the Signature Method in Machine Learning*. Retrieved February 10, 2022, from http://arxiv.org/abs/1603.03788

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf

Cooper, S., Di Fava, A., Vivas, C., Marchionni, L., & Ferro, F. (2020). ARI: The Social Assistive Robot and Companion. *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 745–751. https://doi.org/10.1109/RO-MAN47096.2020.9223470

Danielczuk, M., Mousavian, A., Eppner, C., & Fox, D. (2021, March 26). *Object Rearrangement Using Learned Implicit Collision Functions*. arXiv: 2011.10726 [cs]. Retrieved January 31, 2023, from http://arxiv.org/abs/2011.10726

Dann, C., & Brunskill, E. (2015). Sample Complexity of Episodic Fixed-horizon Reinforcement Learning. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, 2818–2826. http://dl.acm.org/citation.cfm?id=2969442.2969555

Deisenroth, M. P., Fox, D., & Rasmussen, C. E. (2015). Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(2), 408–423. https://doi.org/10.1109/TPAMI.2013.218

Deisenroth, M. P., & Rasmussen, C. E. (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. *Proceedings of the 28th International Conference on Machine Learning, (ICML)*, 465–472.

Dhar, A., & Bhasin, S. (2018). Novel Adaptive MPC Design for Uncertain MIMO Discrete-time LTI Systems with Input Constraints. *2018 European Control Conference (ECC)*, 319–324. https://doi.org/10.23919/ECC.2018.8550217

Di Cairano, S., & Kolmanovsky, I. V. (2019). Automotive applications of model predictive control. In S. V. Raković & W. S. Levine (Eds.), *Handbook of model predictive control* (pp. 493–527). Springer International Publishing. https://doi.org/10.1007/978-3-319-77489-3_21

Diggle, P. J., & Gratton, R. J. (1984). Monte Carlo Methods of Inference for Implicit Statistical Models. *Journal of the Royal Statistical Society. Series B (Methodological)*, *46*(2), 193–227. Retrieved June 27, 2023, from https://www.jstor.org/stable/2345504

Ding, Y., Wang, L., Li, Y., & Li, D. (2018). Model predictive control and its application in agriculture: A review. *Computers and Electronics in Agriculture*, *151*, 104–117. https://doi.org/10.1016/j.compag.2018.06.004

Dong, J., Mukadam, M., Dellaert, F., & Boots, B. (2016). Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. *Robotics: Science and Systems XII*, 1–9. https://doi.org/10.15607/RSS.2016.XII.001

Doucet, A. (2001). *Sequential Monte Carlo Methods in Practice*. Retrieved November 19, 2020, from https://doi.org/10.1007/978-1-4757-3437-9
OCLC: 863947942.

Eaton, J. W., & Rawlings, J. B. (1991). Model Predictive Control of Chemical Processes. *1991 American Control Conference*, 1790–1795. https://doi.org/10.23919/ACC.1991.4791693

Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., & Todorov, E. (2013). An integrated system for real-time model predictive control of humanoid robots. *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 292–299. https://doi.org/10.1109/HUMANOIDS.2013.7029990

Fan, D., Agha, A., & Theodorou, E. (2020). Deep Learning Tubes for Tube MPC. *Robotics: Science and Systems XVI*. https://doi.org/10.15607/RSS.2020.XVI.087

Fermanian, A. (2021). Embedding and learning with signatures. *Comput. Stat. Data Anal.*, *157*, 107148. http://arxiv.org/abs/1911.13211

Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., & Tomlin, C. J. (2019). A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. *IEEE Trans. Automat. Contr.*, *64*(7), 2737–2752. https://doi.org/10.1109/TAC.2018.2876389

Friston, K. (2010). The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, *11*(2), 127–138. https://doi.org/10.1038/nrn2787

Gammell, J. D., & Strub, M. P. (2021). Asymptotically Optimal Sampling-Based Motion Planning Methods. *Annu. Rev. Control Robot. Auton. Syst.*, *4*(1), 295–318. https://doi.org/10.1146/annurev-control-061920-093753

Ganegedara, T., Ott, L., & Ramos, F. (2016). Online Adaptation of Deep Architectures with Reinforcement Learning. *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, 577–585. https://doi.org/10.3233/978-1-61499-672-9-577

Gharbia, M., Chang-Richards, A., Lu, Y., Zhong, R. Y., & Li, H. (2020). Robotic technologies for on-site building construction: A systematic review. *Journal of Building Engineering*, *32*, 101584. https://doi.org/10.1016/j.jobe.2020.101584

Gillani, N. (2023, April 19). *Will AI ever reach human-level intelligence? We asked five experts*. The Conversation. Retrieved May 1, 2023, from http://theconversation.com/will-ai-ever-reach-human-level-intelligence-we-asked-five-experts-202515

Gonzalez, D., Perez, J., Milanes, V., & Nashashibi, F. (2016). A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Trans. Intell. Transport. Syst.*, *17*(4), 1135–1145. https://doi.org/10.1109/TITS.2015.2498841

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, *23*(1), 34–46. https://doi.org/10.1109/TRO.2006.889486

Hämäläinen, P., Babadi, A., Ma, X., & Lehtinen, J. (2020). PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation. *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6. https://doi.org/10.1109/MLSP49062.2020.9231618

Hambly, B., & Lyons, T. (2010). Uniqueness for the signature of a path of bounded variation and the reduced path group. *Ann. Math.*, *171*(1), 109–167. https://doi.org/10.4007/annals.2010.171.109

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. *CoRR*. https://doi.org/10.48550/arXiv.1604.00772

Hansen, N., Müller, S. D., & Koumoutsakos, P. (2003). Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, *11*(1), 1–18. https://doi.org/10.1162/106365603321828970

Harrison, J., Sharma, A., & Pavone, M. (2020). Meta-learning Priors for Efficient Online Bayesian Regression. In M. Morales, L. Tapia, G. Sánchez-Ante & S. Hutchinson (Eds.), *Algorithmic Foundations of Robotics XIII* (pp. 318–337). Springer International Publishing. https://doi.org/10.1007/978-3-030-44051-0_19

Haugh, M. B. (2021). A Tutorial on Markov Chain Monte-Carlo and Bayesian Modeling. *SSRN Journal*. https://doi.org/10.2139/ssrn.3759243

Heess, N., Tb, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M., & Silver, D. (2017, July 7). *Emergence of Locomotion Behaviours in Rich Environments*. arXiv.org. Retrieved March 31, 2023, from https://arxiv.org/abs/1707.02286v2

Heilmeier, A., Wischnewski, A., Hermansdorfer, L., Betz, J., Lienkamp, M., & Lohmann, B. (2020). Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, *58*(10), 1497–1527. https://doi.org/10.1080/00423114.2019.1631455

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning That Matters. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3207–3214. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669

Houska, B., Ferreau, H. J., & Diehl, M. (2011). ACADO toolkit—An open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, *32*(3), 298–312. https://doi.org/10.1002/oca.939

IEEE Spectrum. (2023). *How to Turn the Lights Back on After a Blackout*. Retrieved June 27, 2023, from https://spectrum.ieee.org/power-grid-failure-lights-on

Intuition Robotics. (2023). *ElliQ, the sidekick for healthier, happier aging*. Retrieved June 27, 2023, from https://elliq.com/

iRobot. (2023). *Roomba*. Retrieved June 27, 2023, from https://www.irobot.com.a u/

Islam, R., Henderson, P., Gomrokchi, M., & Precup, D. (2017). Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *CoRR*, *abs/1708.04133*. http://arxiv.org/abs/1708.04133

Jha, S., & Lincoln, P. (2018). Data Efficient Learning of Robust Control Policies. *56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018, Monticello, IL, USA, October 2-5, 2018*, 856–861. https://doi.o rg/10.1109/ALLERTON.2018.8636072

Jones, M., Baerentzen, J., & Sramek, M. (2006). 3D distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, *12*(4), 581–599. https://doi.org/10.1109/TVCG.2006.56

Julier, S. J., & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, *92*(3), 401–422. https://doi.org/10.1109/JPROC.2 003.823141

Julier, S. (2002). The scaled unscented transformation. *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, 4555–4559 vol.6. https: //doi.org/10.1109/ACC.2002.1025369

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. *2011 IEEE International Conference on Robotics and Automation*, 4569–4574. https://doi.org /10.1109/ICRA.2011.5980280

Kamel, M., Stastny, T., Alexis, K., & Siegwart, R. (2017). Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In A. Koubaa (Ed.), *Robot operating system (ROS): The complete reference (volume 2)* (pp. 3–39). Springer International Publishing. https://doi.org/1 0.1007/978-3-319-54927-9_1

Katz, D. M., Bommarito, M. J., Gao, S., & Arredondo, P. (2023, March 15). *GPT-4 Passes the Bar Exam*. https://doi.org/10.2139/ssrn.4389233

Kavraki, L., Svestka, P., Latombe, J.-C., & Overmars, M. (Aug./1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE*

Trans. Robot. Automat., *12*(4), 566–580. https://doi.org/10.1109/70.508439

Kimeldorf, G. S., & Wahba, G. (1970). A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines. *The Annals of Mathematical Statistics*, *41*(2), 495–502. Retrieved April 12, 2023, from https://www.jstor.org/stable/2239347

King, J., Klingensmith, M., Dellin, C., Dogar, M., Velagapudi, P., Pollard, N., & Srinivasa, S. (2013). Pregrasp Manipulation as Trajectory Optimization. *Robotics: Science and Systems IX*, 1–8. https://doi.org/10.15607/RSS.2013.IX.015

Kiraly, F. J., & Oberhauser, H. (2019). Kernels for Sequentially Ordered Data. *J. Mach. Learn. Res.*, *20*(31), 31:1–31:45. http://jmlr.org/papers/v20/16-314.html

Kirk, D. (2012). *Optimal control theory: An introduction*. Dover Publications. https://books.google.com.au/books?id=onuHoPnZwV4C

Koch, W., Mancuso, R., West, R., & Bestavros, A. (2019). Reinforcement Learning for UAV Attitude Control. *ACM Transactions on Cyber-Physical Systems*, *3*(2), 1–21. https://doi.org/10.1145/3301273

Kozłowski, K., & Pazderski, D. (2004). Modeling and Control of a 4-wheel Skid-steering Mobile Robot. *Int. J. Appl. Math. Comput. Sci.*, *14*(4), 477–496.

Kreyszig, E. (1989). *Introductory functional analysis with applications* (Wiley classics library ed). Wiley.

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., & Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Auton Robot*, *40*(3), 429–455. https://doi.org/10.1007/s10514-015-9479-3

Lambert, A., & Boots, B. (2021, July 11). *Entropy Regularized Motion Planning via Stein Variational Inference*. arXiv: 2107.05146 [cs]. Retrieved October 11, 2022, from http://arxiv.org/abs/2107.05146

Lambert, A., Fishman, A., Fox, D., Boots, B., & Ramos, F. (2020–November 18). Stein Variational Model Predictive Control. *Proceedings of the 2020 Conference on Robot Learning*, *155*, 1278–1297.

LaValle, S. M., & Kuffner, J. J. (2001). Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, *20*(5), 378–400. https://doi.org/10.1177/02783640122067453

LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press. OCLC: ocm65301992.

Lax, P. D. (2014). *Functional Analysis* (Online-ausg). Wiley. OCLC: 934669143.

Leeuwen, P. J., Künsch, H. R., Nerger, L., Potthast, R., & Reich, S. (2019). Particle filters for high-dimensional geoscience applications: A review. *Q.J.R. Meteorol. Soc.*, *145*(723), 2335–2365. https://doi.org/10.1002/qj.3551

Levine, S. (2018, May 20). *Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review*. Retrieved October 31, 2019, from http://arxiv.org/abs/1805.00909

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2015). *End-to-End Training of Deep Visuomotor Policies*. https://doi.org/10.1007/s13398-014-0173-7.2

Levine, S., & Koltun, V. (2013). Guided Policy Search. *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 1–9. http://jmlr.org/proceedings/papers/v28/levine13.html

Lewis, F. L., Vrabie, D., & Vamvoudakis, K. G. (2012). Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers. *IEEE Control Systems Magazine*, *32*(6), 76–105. https://doi.org/10.1109/MCS.2012.2214134

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. http://arxiv.org/abs/1509.02971

Liu, Q., & Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 1–13, Vol. 29). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 1–16. Retrieved January 23, 2023, from https://openreview.net/forum?id=Skq89Scxx

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., & Mordatch, I. (2019). Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control, 1–15. http://arxiv.org/abs/1811.01848

Lu, G., Li, S., Mai, G., Sun, J., Zhu, D., Chai, L., Sun, H., Wang, X., Dai, H., Liu, N., Xu, R., Petti, D., Li, C., Liu, T., & Li, C. (2023, April 12). *AGI for Agriculture*. arXiv.org. Retrieved May 1, 2023, from https://arxiv.org/abs/2304.06136v1

Lyons, T. (2014). Rough paths, Signatures and the modelling of functions on streams. *Proceedings of the International Congress of Mathematicians*. Retrieved July 13, 2022, from http://arxiv.org/abs/1405.4537

Lyons, T. J., Picard, J., & Lévy, T. (2007). *Differential equations driven by rough paths: École d'ete de probabilites de Saint-Flour XXXIV-2004*. Springer.

MacKay, D. J. C. (2019). *Information theory, inference, and learning algorithms* (22nd printing). Cambridge University Press.

Mandlekar, A., Zhu, Y., Garg, A., Booher, J., Spero, M., Tung, A., Gao, J., Emmons, J., Gupta, A., Orbay, E., Savarese, S., & Fei-Fei, L. (2018). *RoboTurk: A Crowdsourcing Platform for Robotic Skill Learning through Imitation* (CoRL). http://arxiv.org/abs/1811.02790

Marinho, Z., Boots, B., Dragan, A., Byravan, A., J. Gordon, G., & Srinivasa, S. (2016). Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces. *Robotics: Science and Systems XII*. https://doi.org/10.15607/RSS.2016.XII.046

Marjoram, P., Molitor, J., Plagnol, V., & Tavaré, S. (2003). Markov chain Monte Carlo without likelihoods. *Proc. Natl. Acad. Sci. U.S.A.*, *100*(26), 15324–15328. https://doi.org/10.1073/pnas.0306899100

Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, *50*(12), 2967–2986. https://doi.org/10.1016/j.automatica.2014.10.128

Mesbah, A. (2016). Stochastic Model Predictive Control: An Overview and Perspectives for Future Research. *IEEE Control Systems Magazine, 36*(6), 30–44. https://doi.org/10.1109/MCS.2016.2602087

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In A. Vedaldi, H. Bischof, T. Brox & J.-M. Frahm (Eds.), *Computer Vision – ECCV 2020* (pp. 405–421). Springer International Publishing. https://doi.org/10.1007/978-3-030-58452-8_24

Mordatch, I., & Todorov, E. (2014). Combining the benefits of function approximation and trajectory optimization. *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*. https://doi.org/10.15607/RSS.2014.X.052

Mukadam, M., Dong, J., Dellaert, F., & Boots, B. (2017). Simultaneous Trajectory Estimation and Planning via Probabilistic Inference. *Robotics: Science and Systems XIII*. https://doi.org/10.15607/RSS.2017.XIII.025

Mukadam, M., Dong, J., Yan, X., Dellaert, F., & Boots, B. (2018). Continuous-time Gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research, 37*(11), 1319–1340. https://doi.org/10.1177/0278364918790369

Munkres, J. R. (2014). *Topology* (2. ed., Pearson new internat. ed). Pearson.

Okada, M., & Taniguchi, T. (2020, October 30–November 1). Variational inference MPC for bayesian model-based reinforcement learning. In L. P. Kaelbling, D. Kragic & K. Sugiura (Eds.), *Proceedings of the conference on robot learning (CoRL)* (pp. 258–272, Vol. 100). PMLR. http://proceedings.mlr.press/v100/okada20a.html

Pan, Y., & Theodorou, E. (2014). Probabilistic Differential Dynamic Programming. *Advances in Neural Information Processing Systems, 27*. Retrieved February 20, 2023, from https://papers.nips.cc/paper/2014/hash/7fec306d1e665bc9c748b5d2b99a6e97-Abstract.html

Pan, Y., Yan, X., Theodorou, E. A., & Boots, B. (2017, August 6–11). Prediction under uncertainty in sparse spectrum Gaussian processes with applications to filtering and control. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the*

34th international conference on machine learning (pp. 2760–2768, Vol. 70). PMLR. http://proceedings.mlr.press/v70/pan17a.html

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810. https://doi.org/10.1109/ICRA.2018.8460528

Polydoros, A. S., & Nalpantidis, L. (2017). Survey of Model-Based Reinforcement Learning: Applications on Robotics. *J Intell Robot Syst*, *86*(2), 153–173. https://doi.org/10.1007/s10846-017-0468-y

Possas, R., Barcelos, L., Oliveira, R., Fox, D., & Ramos, F. (2020). Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement, 8.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., & Feldman, M. W. (1999). Population growth of human Y chromosomes: A study of Y chromosome microsatellites. *Molecular Biology and Evolution*, *16*(12), 1791–1798. https://doi.org/10.1093/oxfordjournals.molbev.a026091

Pulido, M., & van Leeuwen, P. J. (2019). Sequential Monte Carlo with kernel embedded mappings: The mapping particle filter. *Journal of Computational Physics*, *396*, 400–415. https://doi.org/10.1016/j.jcp.2019.06.060

Rakovic, S. V., Kouvaritakis, B., Cannon, M., Panos, C., & Findeisen, R. (2012). Parameterized Tube Model Predictive Control. *IEEE Transactions on Automatic Control*, *57*(11), 2746–2761. https://doi.org/10.1109/TAC.2012.2191174

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. *Proceedings of the 38th International Conference on Machine Learning*, 8821–8831. Retrieved May 1, 2023, from https://proceedings.mlr.press/v139/ramesh21a.html

Ramos, F., & Ott, L. (2016). Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, *35*(14), 1717–1730. https://doi.org/10.1177/0278364916684382

Ramos, F., Possas, R., & Fox, D. (2019). BayesSim: Adaptive Domain Randomization Via Probabilistic Inference for Robotics Simulators. *Proceedings of Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2019.XV.029

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.
OCLC: ocm61285753.

Ratliff, N., Zucker, M., Bagnell, J. A., & Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. *2009 IEEE International Conference on Robotics and Automation*, 489–494. https://doi.org/10.1109/ROBOT.2009.5152817

Recht, B. (2018). A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, *2*(1), annurev-control-053018–023825. https://doi.org/10.1146/annurev-control-053018-023825

Ross, I. M. (2009). *A primer on Pontryagin's principle in optimal control*. Collegiate Publishers.
OCLC: 625106088.

Sakhdari, B., & Azad, N. L. (2018). Adaptive Tube-Based Nonlinear MPC for Economic Autonomous Cruise Control of Plug-In Hybrid Electric Vehicles. *IEEE Transactions on Vehicular Technology*, *67*(12), 11390–11401. https://doi.org/10.1109/TVT.2018.2872654

Salvi, C., Cass, T., Foster, J., Lyons, T., & Yang, W. (2021). The Signature Kernel Is the Solution of a Goursat PDE. *SIAM Journal on Mathematics of Data Science*, *3*(3), 873–899. https://doi.org/10.1137/20M1366794

Schaal, S. (1997). Learning from Demonstration. In M. C. Mozer, M. I. Jordan & T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9* (pp. 1040–1046). MIT Press. Retrieved April 9, 2019, from http://papers.nips.cc/paper/1224-learning-from-demonstration.pdf

Schmerling, E., Leung, K., Vollprecht, W., & Pavone, M. (2018). Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3399–3406. https://doi.org/10.1109/ICRA.2018.8460766

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond* (Reprint.). MIT Press.

Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., & Abbeel, P. (2013). Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. *Robotics: Science and Systems IX*. https://doi.org/10.15607/RSS.2013.IX.031

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., & Moritz, P. (2015). Trust Region Policy Optimization. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 1889–1897. http://jmlr.org/proceedings/papers/v37/schulman15.html

Scott, D. W. (1992, August 17). *Multivariate Density Estimation: Theory, Practice, and Visualization* (1st ed.). Wiley. https://doi.org/10.1002/9780470316849

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. A. (2014). Deterministic Policy Gradient Algorithms. *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 387–395. http://jmlr.org/proceedings/papers/v32/silver14.html

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Springer US. https://doi.org/10.1007/978-1-4899-3324-9

Simchowitz, M., Mania, H., Tu, S., Jordan, M. I., & Recht, B. (2018, February 22). *Learning Without Mixing: Towards A Sharp Analysis of Linear System Identification*. Retrieved July 16, 2019, from http://arxiv.org/abs/1802.08334

Steinwart, I., & Christmann, A. (2008). *Support vector machines* (1st ed). Springer.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (Second edition). The MIT Press.

Tassa, Y., Erez, T., & Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4906–4913. https://doi.org/10.1109/IROS.2012.6386025

Tassa, Y., Mansard, N., & Todorov, E. (2014). Control-limited differential dynamic programming. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 1168–1175. https://doi.org/10.1109/ICRA.2014.6907001

Tevel. (2023). *Flying Autonomous Robots*. Retrieved June 26, 2023, from https://www.tevel-tech.com/

Thangavel, S., Lucia, S., Paulen, R., & Engell, S. (2017). Robust nonlinear model predictive control with reduction of uncertainty via dual control. *2017 21st International Conference on Process Control (PC)*, 48–53. https://doi.org/10.1109/PC.2017.7976187

Thrun, S. (2002, March). *Probabilistic robotics* (Vol. 45). Retrieved June 19, 2023, from https://dl.acm.org/doi/10.1145/504729.504754

Tjanaka, B., Fontaine, M. C., Kalkar, A., & Nikolaidis, S. (2022). Training Diverse High-Dimensional Controllers by Scaling Matrix Adaptation MAP-Annealing. *CoRR*, *abs/2210.02622*. https://doi.org/10.48550/arXiv.2210.02622

Todorov, E., & Weiwei Li. (2005). A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *Proceedings of the 2005, American Control Conference, 2005.*, 300–306. https://doi.org/10.1109/ACC.2005.1469949

Trevelyan, J., Hamel, W. R., & Kang, S.-C. (2016). Robotics in Hazardous Applications. In B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics* (pp. 1521–1548). Springer International Publishing. https://doi.org/10.1007/978-3-319-32552-1_58

UC Berkeley D-Lab. (2015). *High Frequency Trading*. Retrieved June 27, 2023, from https://datasci.berkeley.edu/project/high-frequency-trading

Valassakis, E., Ding, Z., & Johns, E. (2020). Crossing The Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics. Retrieved November 19, 2020, from http://arxiv.org/abs/2008.06686

van der Merwe, R. (2004, January). *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models* [Doctoral dissertation, Oregon Health & Science University]. Beaverton, Oregon.

Wabersich, K. P., & Zeilinger, M. (2020). Bayesian model predictive control: Efficient model exploration and regret bounds using posterior sampling. *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 455–464. Retrieved February 20, 2023, from https://proceedings.mlr.press/v120/wabersich20a.html

Wagener, N., Cheng, C.-A., Sacks, J., & Boots, B. (2019). An Online Learning Approach to Model Predictive Control.

Wahrmann, D., Hildebrandt, A.-C., Wittmann, R., Sygulla, F., Rixen, D., & Buschmann, T. (2016). Fast object approximation for real-time 3D obstacle avoidance with biped robots. *2016 IEEE International Conference on Ad-*

vanced Intelligent Mechatronics (AIM), 38–45. https://doi.org/10.1109/AIM.2016.7576740

Watson, J., Abdulsamad, H., & Peters, J. (2020). Stochastic Optimal Control as Approximate Input Inference. *Proceedings of the Conference on Robot Learning*, 697–716. Retrieved February 20, 2023, from https://proceedings.mlr.press/v100/watson20a.html

Williams, G., Drews, P., Goldfain, B., Rehg, J. M., & Theodorou, E. A. (2018). Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, *34*(6), 1603–1622. https://doi.org/10.1109/TRO.2018.2865891

Williams, G., Aldrich, A., & Theodorou, E. A. (2017). Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics*, *40*(2), 344–357. https://doi.org/10.2514/1.G001921

Williams, G., Drews, P., Goldfain, B., Rehg, J. M., & Theodorou, E. (2016). Aggressive driving with model predictive path integral control. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1433–1440. https://doi.org/10.1109/ICRA.2016.7487277

Williams, G., Goldfain, B., Drews, P., Saigol, K., Rehg, J., & Theodorou, E. (2018). Robust Sampling Based Model Predictive Control with Sparse Objective Information. *Robotics: Science and Systems XIV*. https://doi.org/10.15607/RSS.2018.XIV.042

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., & Theodorou, E. (2017). Information theoretic MPC for model-based reinforcement learning. *2017 IEEE International Conference on Robotics and Automation*, 1714–1721. https://doi.org/10.1109/ICRA.2017.7989202

Wittmann, R., Hildebrandt, A.-C., Wahrmann, D., Rixen, D., & Buschmann, T. (2015). State estimation for biped robots using multibody dynamics. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2166–2172. https://doi.org/10.1109/IROS.2015.7353667

Yang, W., Lyons, T., Ni, H., Schmid, C., & Jin, L. (2017, July 13). *Developing the Path Signature Methodology and its Application to Landmark-based Human Action Recognition* (1). arXiv: 1707.03993 [cs]. Retrieved November 22, 2022, from http://arxiv.org/abs/1707.03993

Yi, J., Wang, H., Zhang, J., Song, D., Jayasuriya, S., & Liu, J. (2009). Kinematic Modeling and Analysis of Skid-Steered Mobile Robots With Applications to Low-Cost Inertial-Measurement-Unit-Based Motion Estimation. *IEEE Transactions on Robotics*, *25*(5), 1087–1097. https://doi.org/10.1109/TRO.2009.2026506

Yu, H., & Chen, Y. (2022). A Gaussian variational inference approach to motion planning. *Computing Research Repository*, *abs/2209.05655*. https://doi.org/10.48550/arXiv.2209.05655

Zanon, M., Frasch, J., Vukov, M., Sager, S., & Diehl, M. (2014, March). Model predictive control of autonomous vehicles. In *Lecture Notes in Control and Information Sciences* (Vol. 455). https://doi.org/10.1007/978-3-319-05371-4_3

Zhuo, J., Liu, C., Shi, J., Zhu, J., Chen, N., & Zhang, B. (2018). Message Passing Stein Variational Gradient Descent. *ICML*, 10.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, *32*(9-10), 1164–1193. https://doi.org/10.1177/0278364913488805