Stochastic Hybrid Model Predictive Control using Gaussian Processes for Systems with Piecewise Residual Dynamics

by

Leroy Joel D'Souza

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

Abstract

Due to their ability to model complex functions with uncertainty estimates, Gaussian Processes (GPs) have found widespread use to learn residual dynamics that account for the mismatch between the nominal system model and the true underlying system dynamics. However, existing literature using GPs for control considers true residual dynamics that are not piecewise functions differing over regions of the state space. Trying to approximate such piecewise residuals by a single learnt GP-based residual model leads to inaccurate dynamics predictions across a horizon when working with a Model Predictive Controller (MPC) resulting in poor closed-loop performance and constraint violation above a specified threshold.

In this thesis, I first propose the construction of a hybrid GP learnt from data for systems operating in environments with piecewise residuals. The procedure to embed such a model into a Chance-Constrained Model Predictive Controller (CC-MPC) is formulated as a Mixed Integer Nonlinear Program (MINLP) by introducing a set of constrained discrete variables. This allows a more accurate propagation of dynamics across a horizon enabled by switching between different modes of the learnt hybrid residual at each timestep. Through numerical studies, I demonstrate how the proposed controller outperforms a baseline CC-MPC (using only a single GP model to capture residual dynamics) in terms of both closed-loop performance and chance constraint satisfaction.

In general, it is not possible to solve these MINLPs while meeting the control input frequency demanded by real-time applications like vehicle control. I then propose an algorithm to convert the MINLP problem into a parametrized Nonlinear Program (NLP) using a hierarchical planner-controller approach while also specifying other parts of the optimization that can be relegated to an offline computation block. Simulations show that the proposed approximate NLP solution displays a reasonable level of performance when compared to the MINLP solution.

The previous problems involve the assumption that the “regions” where a particular mode of the residual is active are known in advance. I consider the problem of relaxing this assumption to allow transfer to new environments in which it is desired to execute a repetitive task. This is done by leveraging the learnt residual to identify region information and train a classifier that can be utilized for predictions for future runs. Finally, I also propose an approach to improve the classifier training dataset accuracy using an optimization-based approach based on maximizing distances between distributions predicted by different modes of the hybrid residual.

Acknowledgements

I am extremely grateful to my supervisor, Yash Pant, for providing me the opportunity to work with the CL2 group at Waterloo and for the many suggestions and fruitful discussions that have never gone unappreciated. I would also like to thank friends across the graduate sphere in the ECE, SYDE and MME departments at Waterloo for their help and support.

Dedication

I dedicate this thesis to my parents, Edward and Rina, without whose unwavering love, support and guidance, none of this would ever have been possible.

Table of Contents

Author's Declaration **ii**

Abstract **iii**

Acknowledgements **iv**

Dedication **v**

List of Figures **ix**

List of Tables **xiv**

List of Abbreviations **xvi**

1 Introduction **1**

1.1 Literature Review 2

1.1.1 Data-driven control 2

1.1.2 Hybrid MPC 5

1.1.3 Stochastic MPC 5

1.2 Contributions 6

2 Background **7**
2.1 Gaussian Processes - An Overview 7
2.2 Introduction to MPC 11
2.2.1 Tractability of Gaussian Process-based MPC 12
2.2.2 Stochastic Chance-Constrained MPC 13

3 Designing a Hybrid CC-MPC Controller **17**
3.1 Introduction 17
3.1.1 Motivating Example 18
3.2 Problem Statement 19
3.3 Learning Hybrid GP residual models 22
3.4 Demonstrating the benefits of hybrid GP models 24
3.4.1 Effects of discontinuities in residuals. 24
3.4.2 Addressing issues with learnt lengthscales. 27
3.4.3 Residual dynamics with overlapping input spaces. 30
3.5 Designing an SMPC Controller for a model with hybrid piece-wise dynamics 31
3.5.1 Control Policy. 31
3.5.2 Propagating residual dynamics across a horizon. 31
3.5.3 Shrunk chance-constrained sets. 33
3.5.4 Big-M formulation for the hybrid residual model. 33
3.5.5 Cost Function. 36
3.5.6 Hybrid MPC Formulation 36
3.6 Improving computational speed of a hybrid GP-MPC controller. 37
3.6.1 Cause of slow MINLP solve times. 38
3.6.2 Constructing a parametrized hybrid MPC controller. 38
3.6.3 Planning considerations. 42
3.7 Numerical Simulations 44
3.7.1 Mixed-Integer GP-MPC Controller. 44
3.7.2 Hybrid NLP GP-MPC Controller. 49
3.8 Conclusion 54

4 Identifying region locations in unseen environments. 55

4.1 Motivating Example. 55

4.2 Problem Setup and Statement 56

4.3 Methodology 60

4.3.1 Training a classifier on a dataset of soft label vectors. 60

4.3.2 Soft vs. Hard label training datasets 64

4.3.3 Terrain mapping planner-controller architecture 67

4.4 Data-efficient mapping of unknown environments. 68

4.4.1 Incorporating optimization results into the nominal controller. . . . 75

4.5 Results 76

4.5.1 Data-efficient mapping results 77

4.5.2 Iterative terrain mapping results 80

4.6 Conclusion 82

5 Conclusion 84

5.1 Limitations and Future Research Directions 85

5.1.1 MINLP to NLP conversion. 85

5.1.2 Terrain mapping-based control. 86

5.1.3 Online learning and control. 87

5.1.4 Real-World Implementation 88

References 89

APPENDICES 103

A Dynamics model for a 2-D quadrotor. 104

Glossary 106

List of Figures

2.1 (a) and (c) depict examples of a Gaussian Process with a mean function and covariance function (illustrated using symmetric 2-σ confidence intervals). Data samples are depicted in red and the ground truth mean is plotted in green. Note that (c) has a higher frequency mean than (a). (b) and (d) show arbitrary function realizations sampled from the GP across the domain of the input space. (d) clearly extrapolates less from training samples as compared to (b) due to the smaller learnt lengthscale and hence has higher uncertainty in regions of low training data density. 8

2.2 A closed-loop MPC controller where the first optimal control action is applied to the plant. The measured next state is passed back to the controller, re-parameterizing the initial value constraint and shifting the reference trajectory after which the optimization is re-solved. 12

2.3 A visual demonstration of the workings of gradient-based optimizers 13

2.4 Inspired by [23]. A visual demonstration of how the mean of the state affects constraint satisfaction for a given variance term. For both distributions, the means μ_1^x, μ_2^x satisfy the constraint $-x \leq -a (x \geq a)$. For the distribution in green, the left-sided tail does not exceed the specified violation probability while the distribution in yellow does. Thus the state mean plays a big role in constraint satisfaction. 15

3.1 A wheeled robot driving over different terrains. The contact dynamics of the tyres will differ across regions and also within the regions, making it difficult for a single model to accurately capture the robot dynamics across the entire workspace. 18

3.2 Plots of the residual magnitude (y-axis) vs. the single state variable input to the GP (x-axis) (a) Plot of the true underlying piece-wise residual dynamics and the samples generated from it to be used to train the GP. (b) Plot of the single GP and the learnt covariance function. (c) Plot of the independent region-specific trained GPs visualized over the entire state space. (d) Plot of hybrid piece-wise trained GP with mean and variance computed as defined in (3.8b). (e) Mean function absolute error for single GP approach over state domain (f) Mean function absolute error for hybrid GP approach over state domain 26

3.3 (a) Shows a piecewise residual with a significant discontinuity between its modes. (b) Shows the output after training a single GP model. The short learnt lengthscale introduces noise into the mean function and causes uncertainty to rise sharply in areas of low data density. (c) Shows the output from a hybrid GP model that is able to extrapolate the high lengthscale piece-wise mode well in the absence of data. 28

3.4 (a) True sinusoidal residual underlying dynamics with a high frequency sinusoid in region 2. (b) A learnt single GP model where hyperparameter optimization has aligned with the parameters in region 1. (c) A learnt single GP model where hyperparameter optimization has aligned with the parameters in region 2. (d) Learnt hybrid GP model with 2 modes that remedies the issues of the single GP models. 29

3.5 Illustrative example. (a) Visual mapping of different terrains across the workspace to the corresponding modes of the residual model. (b) A single GP model which learns a very noisy function due to the low correlation between the different modes across the input space. (c) The hybrid GP model that accurately learns the piecewise residual dynamics with an overlapping input space. 30

3.6 Optimizing over the space of trajectories to get from the start position (red) to the destination (green) leads to different assignments to the discrete δ variables at each timestep. These assignments can be element-wise multiplied with the vector of residual outputs from the hybrid GP model. As a simple example, consider the third timestep of the purple path and assume it belongs to the 2nd region. In this case, since region 2 is the active mode, the residual term from that region is selected for the dynamics equation. . 34

3.7 (a) An example of RRT outputs over multiple runs trying to find paths between a set of high-level waypoints. (b) An example of a spline-interpolated trajectory generated from the waypoints output by RRT. This trajectory can then be discretized as desired to obtain a reference trajectory for the online controller to track. 43

3.8 Residual dynamics function (g) used for all 3 cases. 45

3.9 Region 1 (peach), region 2 (cyan) and region 3 (green). Baseline GP-MPC (left) and Proposed Hybrid Piece-wise GP-MPC (right) closed loop trajectories for the stabilization task. 46

3.10 Baseline GP-MPC (left) and Proposed Hybrid Piece-wise GP-MPC (right) closed loop trajectories showing constraint violations while tracking a set-point at the state-space boundary. 47

3.11 Baseline GP-MPC (left) and Proposed Hybrid Piece-wise GP-MPC (right) closed loop trajectories for tracking a set-point at the boundary of the state space. 48

3.12 A plot depicting the (a) partition of the workspace into 3 regions (b) the residual magnitude and noise for each mode of the piece-wise dynamics and the corresponding region they are active in. 50

3.13 A plot depicting an example of the closed-loop $x-z$ trajectories obtained for each controller i.e., nominal MPC, nominal MPC with mean residual dynamics, hybrid GP-MPC with p_x=0.99, 0.9 and hybrid GP-MPC with $p_x = 0.9$ and "relaxed" shrinking constraints. 51

4.1 A wheeled robot with access to a hybrid residual model. Can it leverage known information along with measurements to build a "map" of how the terrains are distributed across the environment it is operating in? 56

4.2 A directed graph showing dependence relations between the different variables affecting the residual magnitude. 58

4.3 A visual depiction of the process of computing likelihoods that a particular measured residual was generated by one of the learnt modes of the hybrid model for a given input. 61

4.4 (a) Trained hybrid GP model. (b) Ground truth regions spread across workspace. (c)/(d) Example of uniform random sampling and biased sampling across G_{in} respectively. 64

4.5 Plots depicting the qualitative outputs on a test dataset after training classifiers using the soft (b)/(d) and hard label (a)/(c) approaches when working with a random sampled dataset. (a)/(b) uses 200 training samples and (c)/(d) uses 5000 training samples. Row 1 corresponds to the argmax predictions. Row 2, 3 and 4 correspond to classifier logit outputs for each region after applying the softmax operator. 65

4.6 Qualitative outputs on a test dataset after training classifiers using the soft (b) and hard label (c) approaches when working with a biased dataset. . . 66

4.7 A graph showing accuracy with error-bars of both (a) soft and (b) hard label trained classifiers as a function of the number of training samples ([100, 200, 500, 1000, 3000]). 67

4.8 Terrain locations and corresponding modes of the piecewise residual dynamics active in each of them. 76

4.9 (a) / (c) Output from the simulated annealing optimization with num_runs = 20, top_k=10, redundancy_radius=0.45, using a Bhattacharya distance metric that tries to maximize / minimize, respectively, the sum of the pairwise distances between the elements of $\mathcal{N}_{set}$. (b) / (d) Output from the same optimization but with redundancy radius functionality excluded and solutions within a 0.1 radius of each other being pruned. 77

4.10 A visualization of the cost function that results after incorporating the exploration cost C_{expl} generated using the array x_{opt} produced by finding points of maximum distance between the distributions in $\mathcal{N}_{\text{set}}$. (a) depicts the plot when the redundancy is included and (b) shows the plot when redundancy is not included. 78

4.11 A qualitative comparison of the closed-loop trajectories obtained for various assignments parameters in the exploration-based optimization framework. Row (a) shows $x - z$ trajectories when the redundancy radius is excluded and (b) shows trajectories where they are included. 79

4.12 Plots demonstrating the training procedure for the iterative mapping-based classifier and the benefits it yields on closed-loop trajectories over repeated runs on the repetitive tracking task. 81

4.13 Plots demonstrating the possible application of the framework to iterative safe exploration under uncertainty. 82

A.1 [146] A 2-D quadrotor model subject to 2 input thrusts T_1, T_2 with tilt angle θ operating in the $x - z$ plane. 104

List of Tables

3.1 Comparison of absolute errors in the mean (e_μ) and covariance (e_σ) functions for the hybrid and single GP models over the domains X, R_1, R_2. Errors computed over 30 runs using 400 samples to train the model and 120 values across X for verification to generate the error terms (3.12). 25

3.2 Mean and variance of the closed loop costs between the baseline and proposed approach over 20 simulation runs for the stabilization task. A lower cost implies better control performance. 46

3.3 Averages ($\mu_{\#viol}, V_{ext}(\mu)$) and standard deviation ($\sigma_{\#viol}, V_{ext}(\sigma)$) of the number and extent of state constraint violations with average C.L. cost (μ^c_{cl}) for the baseline and proposed approaches. Computed over 20 simulation runs. 48

3.4 Mean and variance of the closed loop costs between the baseline and proposed approach over 20 simulation runs for the tracking task. Lower cost represents better control performance. 49

3.5 Empirically measured average constraint violation for the closed-loop ($\hat{\delta}_x$) trajectory along with mean and variance of the closed-loop cost. Results in red violate the chance constraints and those highlighted in bold indicate they are the best in comparison to other approaches. Computed over 50 simulation runs. 52

3.6 Run-time statistics comparison between a nominal MPC controller and the proposed parametrized NLP GP-MPC controller computed over 25 runs. μ_{run}, σ_{run} denotes the run-time mean and variance respectively. 53

4.1 A quantitative comparison between the training dataset accuracy generated by various parameters in Algorithm 5. $\alpha = 0$ indicated no exploration and Red. radius $= 0$ indicates a deactivation of the redundancy radius feature. μ_{ce} indicates the average cross-entropy loss per sample computed between the generated soft label and the provided ground truth. μ_{argmax} is the % of soft label that match the ground truth after applying the argmax operator to them. 78

4.2 A comparison of closed-loop performance over repeated runs of the system on the repetitive tracking task. μ^c_{cl} denotes the average closed-loop cost for a given run number and the +/- signify the minimum and maximum deviations from the mean computed over 15 simulations. 80

List of Abbreviations

ARD Automatic Relevance Determination 9

BLR Bayesian Linear Regression 4

BNN Bayesian Neural Network 4

CC-MPC Chance-Constrained Model Predictive Controller iii, 17

CDF Cumulative Distribution Function 16, 33, 51

CVaR Conditional Value-at-Risk 6

DARE Discrete Algebraic Riccati Equation 36

DD-MPC Data-Driven MPC 4

GMM Gaussian Mixture Model 6

GP Gaussian Process iii, x, 1–4, 6–10, 12–14, 17, 19, 22–25, 27–35, 38, 40, 44–46, 52, 53, 60–62, 64, 65, 72–74, 76, 82, 84, 86–88

IL Imitation Learning 4

LSTM Long short-term memory 4

MINLP Mixed Integer Nonlinear Program iii, 17, 36–38, 52, 54, 84, 85

ML Machine Learning 1, 5, 12

MLE Maximum Likelihood Estimation 9

MPC Model Predictive Controller iii, 1, 4–6, 11, 17, 30, 36–38, 42, 44, 45, 50, 52–54, 69, 72, 84, 87, 106

MRAC Model Reference Adaptive Control 1

NLP Nonlinear Program iii, 11, 38, 41, 44, 49, 52–54, 65, 68, 84, 85

NN Neural Network 1, 3, 4, 62–64, 67, 76, 83, 86

OOD Out of Distribution 83

PWM Pulse Width Modulation 105

RBF Radial Basis Function 4

ReLU Rectified Linear Unit 76

RL Reinforcement Learning 3, 4

RNN Recurrent Neural Network 4

SE Squared Exponential 7, 9, 24

TCN Temporal-convolutional 4

Chapter 1

Introduction

Dealing with issues stemming from unknown system dynamics has been a long-standing challenge in the domain of control theory. Two main paradigms for addressing these have revolved around adaptive control [83, 6] that aims to enable system performance in the presence of uncertain and/or time-varying plant parameters (e.g., Model Reference Adaptive Control (MRAC) [149], $\mathcal{L}1$ [69]) and robust control [104] that tries to account for worst-case disturbances that a system might be subject to. Advances in computing hardware and optimization theory have caused optimal control methods like MPC [119, 54] to become more common in various control applications and this will be the control method of focus in this thesis.

Machine Learning (ML) methods have been applied to various parts of the autonomy stack for decades now, most notably perception tasks. Off-late applications of data-driven methods for control have started gaining traction in several of the aforementioned control paradigms [29, 63, 38, 68, 17]. While it is tempting to draw on the significant body of work relating to Neural Networks (NNs), practical control applications might deal with comparatively much smaller datasets than other typical NN applications such as trajectory prediction and object classification. When dealing with safety-critical tasks, it is important to take into account uncertainty in the learnt model in regions of low data density while also considering that the true underlying function generating the dataset is not necessarily deterministic but rather stochastic due to several sources of noise (e.g., measurement/sensor, process) that one encounters in practice. As a result, common methods used to learn models for control in the literature utilize Bayesian approaches such as GPs [61].

One paradigm that has gained popularity [61, 13] in recent literature learns unmodelled dynamics from data to be used in addition to nominal models learnt from first principles.

For this reason, I will refer to these learnt dynamics as residual dynamics for the remainder of this thesis. A limitation of these approaches is that they consider a single (uni-modal) GP model when learning model mismatch. This involves the implicit assumption that most datasets for learning-based models are generated from environments that are homogeneous and static in nature i.e., where the true underlying model does not change during and between data collection runs. For example, when learning residual dynamics for a racing car it is assumed that the data collected from the first run would be generated from the same dynamics model as that of the n^{th} run. However, there could be changes to the model either due to changes in the environment (e.g., it started to rain affecting the frictional coefficient of the track) or due to changes in the system (e.g., wear-and-tire of the tyres).

[34] provides a definitive example of how this can lead to overfitting for a simple bipedal walker robot when learning a Bayesian residual model during the task of walking in a circle. Due to the symmetric nature of the system it was expected that the residual model would be the same for both legs. Instead, they turned out to be phase shifted by 5 degrees from each other because of the circular path constraint. Using the same model to attempt to execute other gait patterns might lead to incorrect behaviour as a result of this overfitting, demonstrating that these learnt residuals might inadvertently overfit to a particular task being performed or environment that the system is operating in. This can work well when it is desired to optimize performance on a single unchanging task in a single static environment where one intends to perform repetitive tasks. However, when trying instead to design controllers that generalize to different scenarios, this approach might not perform well.

From a safety perspective, it is beneficial to design controllers using a hybrid model that consolidates information learnt from multiple scenarios and also identifies which scenario-specific information to leverage when, particularly when some scenarios require increased caution than others.

In this thesis, I aim to take steps towards addressing the problem of working with hybrid models capable of encapsulating information useful for several scenarios/tasks.

1.1 Literature Review

1.1.1 Data-driven control

Review by task.

Over the past decade there have been many works that address the problem of learning-based control schemes to improve control performance while adhering to problem-specific notions of safety [29, 67, 107]. There are several tasks of interest in the literature and this section discusses a non-exhaustive list of them.

Efficient exploration for learning Several works in the literature try to address the problem of efficient online learning of system dynamics under uncertainty. Some methods utilize Bayesian optimization approaches and acquisition functions [34] to maximize the probability of improving the model. Others deal with providing robust control guarantees during the learning procedure which is of particular use in highly safety-critical systems where the system model can be extremely uncertain outside of a certain range of operating conditions [19, 81].

Piecewise (hybrid) model identification There has been work that deals with identifying hybrid models from data. The problem under consideration here involves assigning labels to samples in a dataset that identifies them as being generated by a particular mode of the underlying piecewise model. [50, 14] identify piecewise linear models combining regression and clustering based approaches for data with both continuous quantities and discrete labels. When it comes to dealing with models under uncertainty, [90] learns modes of a hybrid GP models under changing system conditions using Bayesian non-parametric clustering approaches. In this thesis, it is assumed that the data available at our disposal already has label assignments to various modes for the purpose of hybrid model learning.

Learning quantities to enforce guarantees Many Reinforcement Learning (RL)-based controllers [57, 118] use NNs to demonstrate significant *empirical* boosts in performance. However, there exists a significant body of work that tries to use NN-based methods while retaining the guarantees provided by control theory. Many of these approaches involve learning Lyapunov functions to guarantee stability e.g., [37] which uses NNs with smooth non-linear activation functions (e.g., tanh) for which Lie derivatives can be computed. [138, 12] tries to learn NNs by optimizing over the space of models satisfying certain apriori specified Lipschitz bounds in order to demonstrate robustness to adversarial perturbations. [58, 112] use NN-based adaptive control methods along with Lyapunov barrier functions [4] to guarantee properties like tracking error convergence.

Operating under new conditions The work in this thesis serves to demonstrate improved performance benefits on tasks for which datasets are already present to train learning-based models on. In constrast, there are works in the literature that leverage existing data-driven models to operate in previously unseen conditions with possibly new dynamics. These often involve using fuzzy methods [143] or simply learning a point-wise weighted computation of the existing models [96].

Review by technique.

There have been many techniques used in the literature for data-driven control and this section covers a non-exhaustive list of them. Some techniques do not learn dynamics from data but rather iteratively construct safe sets and improved cost metrics [114]. In contrast, this section will focus on approaches that use data to build more accurate dynamics models. [141, 30] highlights a learning-based approximation to the Koopman operator that derives from dynamic mode decomposition. [72] deals with identification of sparse nonlinear models using regression with a library of candidate terms yielding a model that is easily interpreted. Other classical approaches involves using auto-regressive models to learn dynamics [2].

When it comes to NN-based approaches, several models have been used in the literature. These include single [70] and multi-layered [115, 85] NNs, Recurrent Neural Networks (RNNs) [151, 145, 106], Radial Basis Function (RBF) networks [116] and more recently, temporal approaches such as Long short-term memory (LSTM) networks [71] and NNs using Temporal-convolutional (TCN) [99] encoder blocks [13].

In terms of bayesian modelling approaches, GPs [61], Bayesian Linear Regression (BLR) [91] and more recently Bayesian Neural Networks (BNNs) [11] have been used in the literature. In this thesis, I choose to focus on GPs which has shown promise in recently published works [73, 61, 45, 38]

While RL and Imitation Learning (IL)-based data-driven control approaches have started to become more popular in the literature, they are not included in this review due to the sheer breadth of research done in this domain as covered by [80, 111, 29].

GP-based constrained optimal control. When considering performance, [129] uses GP-MPC to learn residual aerodynamic effects to improve tracking performance while considering input limitations. [100] applies a learnt GP model to a vision-based control system for a mobile robot to also boost performance on the path-tracking task. In terms of safety, [28] proposes a novel offline sampling-based approach to constraint tightening to allow for chance-constraint constraint satisfaction online. [61] proposes a method for online shrinking of constraint sets to solve the same problem and applies it to an autonomous racing setup to improve performance while maintaining safety.

Review by domain applications.

Since the applications of data-driven methods to general control paradigms are extremely vast, I limit my consideration to MPC-based approaches in this section. Data-Driven MPC (DD-MPC) has been used for power electronics and motor drive control [150], energy optimization [122, 27, 87], tracking tasks for robotic arms [94, 36], agriculture [33], racing [114, 61], mechanical systems such as turbines and blast furnaces [148, 143], flight control [72], marine vessels [144, 59] and manufacturing across different domains [142].

1.1.2 Hybrid MPC

In the controls literature, there are several meanings for the term "hybrid" control. Some refer to controllers that combine physics-based models with ML-based ones and these are also referred to as "gray-box" approaches [10, 27, 87]. Less frequently, this term is also used to refer to schemes that use multiple controllers in parallel with some way to either select between or combine the control signals they produce [44]. In this thesis, however, the term hybrid will be used to denote systems that deal with both continuous and *discrete* variables [16, 15, 35] (although the proposed approach does also come under the "gray-box" category). A lot of work has been done in the past regarding hybrid system control [24, 128, 32] with particular applications to aircrafts [95] and traffic control [42, 125], and highly complex systems dynamical systems like dextrous hands. Some hybrid systems combine several structures together to achieve greater functionality e.g., [108] builds a robot capable of both walking and flying. The hybrid nature of the system is intrinsic to its design. However, the hybrid systems considered in this thesis are those where the discrete variables that define hybrid characteristics of the system are related to variables linked to the environment. An example of this would be a mobile robot operating on rocky surfaces, slippery ice or asphalt depending on where it is in the workspace. To the best of my knowledge, these models have not been considered in the literature for robotics and bayesian learning applications up until recently. In terms of the hybrid problem statement, [96], developed in parallel with the approach presented here, is closest to ours although they do not consider a chance-constrained formulation for the improved safety benefits as has been done in this thesis.

1.1.3 Stochastic MPC

Unlike robust control that handles for worst case disturbances, stochastic (chance-constrained) control approaches accounts for distributions over disturbances particularly ones that have infinite support. There is a vast body of literature that addresses the problem of stochastic chance-constrained MPC for both linear and nonlinear systems [1, 92].

When it comes to uncertainty propagation over a horizon for nonlinear systems, there have been several approaches proposed in the literature to model these distributions. These include techniques reliant on approximation by dynamic linearization [55], sampling-based approaches like MCMC [74, 28, 133], Gaussian Mixture Models (GMMs) and polynomial chaos expansions (common in the spacecraft domain) [134].

In this thesis, I consider approaches that deal with approximation approaches that maintain a time-varying Gaussian uncertainty distribution over the open-loop horizon. This allows leveraging work that provides deterministic reformulations of probabilistic chance-constraints involving Gaussian random variables. Several methods have been proposed to achieve this including conservative ellipsoidal bounds [131], computationally intensive sampling approaches [22] and Conditional Value-at-Risk (CVaR) approaches [132]. However in this thesis, I focus on approaches that use Boole's inequality to convert joint chance constraints into a collection of individual chance constraints. There are several variants of these, most notably the ones that consider an optimization-based risk allocation [98]. However, for simplicity, I will consider a static apriori determined risk allocation scheme which can be more conservative [23, 61].

1.2 Contributions

In Chapter 3, I introduce the problem of controlling a robotic system in an environment where the dynamics model the system is subject to varies as a function of the state e.g., a mobile robot driving in an environment composed of multiple terrains each with their own frictional coefficient. I construct a procedure to learn a hybrid GP model and then embed it in an MPC control framework capable of handling for such a model. Next, I propose an algorithm capable of addressing limitations in the practicality of such controllers that result from significant computational complexity due to the way the optimization problem is formulated.

In Chapter 4, I relax a significant assumption that involves knowing the location of the different regions that divide the state space and consequently determine the active mode of the hybrid model. I propose an algorithm that leverages the information contained in the hybrid model to determine the likelihood with which a measured sample was generated from a particular mode. I then train a classifier capable of utilizing this information to iteratively improve performance on a repetitive task in an unknown environment. Finally, I demonstrate an approach capable of addressing the efficiency of data collection measured in terms of the accuracy of the training dataset.

Chapter 2

Background

2.1 Gaussian Processes - An Overview

GPs are an intuitive extension of mean and variance concepts of Gaussian probability distributions to the function space. In other words, given a set of inputs, a GP outputs a (multi-variate) Gaussian distribution whose mean and covariance is dependent on the inputs provided thus characterizing a distribution over functions as illustrated in Figure 2.1. It is clear, from the large variations in the sampled functions, that there is an increase in uncertainty about what the true underlying function looks like when moving away from regions with high data density.

Given a dataset with samples of the form $(\mathbf{z}, \mathbf{d})$, the GP models this as a multivariate Gaussian of the form

$$p(\mathbf{d} \mid \mathbf{Z}_D) = \mathcal{N}(m(\mathbf{Z}_D), K(\mathbf{Z}_D, \mathbf{Z}_D)) \tag{2.1}$$

where $m(\cdot), K(\cdot, \cdot)$ are the prior mean and covariance functions respectively and $\mathbf{Z}_D$ represents the set of training inputs. The assumption is usually made that the prior mean is constantly zero [55] since it is practically not limiting and simplifies calculations. This assumption will be utilized in this thesis. The form of the covariance kernel can now be considered.

Choice of covariance kernel. The covariances between outputs are characterized by covariance functions/kernels evaluated at their corresponding inputs. There are many kernels used in practice [47] suitable for approximating different types of functions.

The most notable covariance kernel of these is the Squared Exponential (SE) kernel and is used in this thesis. This function belongs to the family of universal approximators [93]

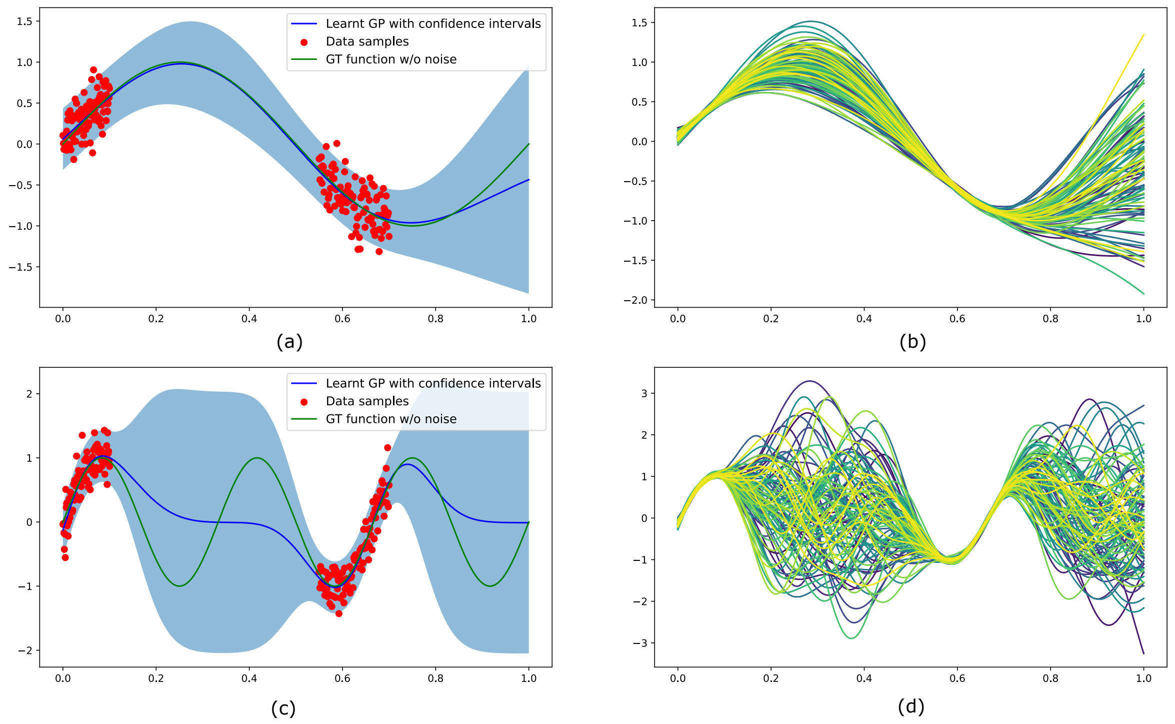


Figure 2.1: (a) and (c) depict examples of a Gaussian Process with a mean function and covariance function (illustrated using symmetric 2-σ confidence intervals). Data samples are depicted in red and the ground truth mean is plotted in green. Note that (c) has a higher frequency mean than (a). (b) and (d) show arbitrary function realizations sampled from the GP across the domain of the input space. (d) clearly extrapolates less from training samples as compared to (b) due to the smaller learnt lengthscale and hence has higher uncertainty in regions of low training data density.

and has the form,

$$C_j(z_1, z_2) = \sigma^2 \exp\left(-\frac{1}{2}(z_1 - z_2)^T \Lambda^{-1}(z_1 - z_2)\right) \tag{2.2}$$

In the context of applications to control systems, the input vectors z_1, z_2 come from the joint state-input space. Defining the state and input constraint sets as X and U this yields $z_1, z_2 \in Z \subseteq X \times U$.

An important hyperparameter is the GP lengthscale. A longer estimated lengthscale for the diagonal entry of a given variable implies that correlations that extend over large distances as shown in Figure 2.1(a)/(b). In contrast, a smaller estimated lengthscale, as in Figure 2.1(c)/(d), implies that the ability to extrapolate away from training data points is limited.

In general, not all variables in Z contribute to variations in the residual term. The SE kernel allows for Automatic Relevance Determination (ARD) based on lengthscale magnitude. A very long lengthscale implies that the function varies minimally across the input domain for that variable. This indicates that it does not play a role in determining the residual magnitude. In this thesis, this procedure will be ignored and the following assumption regarding the GP *inputs* is made.

Assumption 2.1.1. *The variables that are relevant to the space of inputs to the GP are known apriori allowing us to skip the process of identifying them via ARD.*

The following assumption regarding the GP output vectors is also made (applicable when the dimension of the residual vector is > 1).

Assumption 2.1.2. *The residual terms governing different dimensions, $\{1, \ldots, n_d\}$, of the residual vector, d, are independent of each other.*

In light of Assumption 2.1.2, (2.2) can be subscripted by the dimension of the residual vector, j.

$$C_j^{se}(z_1, z_2) = \sigma_j^2 \exp\left(-\frac{1}{2}(z_1 - z_2)^T L_j^{-1}(z_1 - z_2)\right) \tag{2.3}$$

The hyperparameters (i.e., the variance σ_j^2 and the lengthscale L_j) are scalars and $L_j = \Lambda_{jj}$ for Λ described in (2.2).

Remark 2.1.1. *Going forward, the index of the residual vector will no longer be highlighted. A GP model, $\hat{g}$, that outputs a vector will be assumed to benefit from Assumption 2.1.2 and the statements that are true as a result of it.*

When learning real-world functions, there is usually some degree of stochasticity involved and so in addition to the SE kernel parameters, there is also the addition of a Gaussian i.i.d noise hyperparameter σ_n which can be learnt from data. As a result of this, the form of $K(\cdot, \cdot)$ in (2.1) is,

$$K(\mathbf{Z}_D, \mathbf{Z}_D) = C^{se}(\mathbf{Z}_D, \mathbf{Z}_D) + I\sigma_n^2 \tag{2.4}$$

There are many approaches that allow learning the parameters of the covariance kernel used. Maximum Likelihood Estimation (MLE) approaches [55] [140] are most common and they will be utilized in this thesis.

Making predictions. When dealing with a control problem, the datasets collected consist of trajectories with samples of the form (x_k, u_k). To generate a dataset that can be used for GP training, the residuals d_k need to be computed as addressed in Chapter 3. $\tilde{D}$ is used to denote this augmented dataset.

Having learnt parameters for the covariance kernel, the GP can predict a outputs, d, for a new *deterministic* input, z_*. The resulting distribution of d conditioned on the observed data points $\mathbf{Z}_{\tilde{D}}$ and the new input z_* is also Gaussian and can be derived as described in [73],[135] using (2.4).

$$\mu^d(z_*) = K(z_*, \mathbf{Z}_{\tilde{D}})(K(\mathbf{Z}_{\tilde{D}}, \mathbf{Z}_{\tilde{D}}) + I\sigma_n^2)^{-1}\mathbf{d} \tag{2.5a}$$

$$\Sigma^d(z_*) = K(z_*, z_*) - K(z_*, \mathbf{Z}_{\tilde{D}})(K(\mathbf{Z}_{\tilde{D}}, \mathbf{Z}_{\tilde{D}}) + I\sigma_n^2)^{-1}K(\mathbf{Z}_{\tilde{D}}, z_*) \tag{2.5b}$$

This distribution over outputs is denoted as,

$$d = \hat{g}(z_k) \sim \mathcal{N}\Big(\mu^{\hat{g}}(\tilde{D}, z_k), \Sigma^{\hat{g}}(\tilde{D}, z_k)\Big) \tag{2.6}$$

The GP learns non-linear functions $\mu^{\hat{g}}$ and $\Sigma^{\hat{g}}$ to represent the mean and variance respectively of the predicted output. For this thesis, $\mu_k^{\hat{g}}$ will be used as shorthand for $\mu^{\hat{g}}(\tilde{D}, z_k)$, and $\Sigma_k^{\hat{g}}$ can be similarly defined.

As a result of Assumption 2.1.2, separate GPs can be trained for each output dimension of the residual vector, stacking the mean outputs and diagonalizing the variance outputs. Hence, $\mu_k^{\hat{g}} = [\mu_{1,k}^{\hat{g}} \dots \mu_{n_d,k}^{\hat{g}}]^T$ and $\Sigma_k^{\hat{g}} = \text{diag}(\Sigma_{1,k}^{\hat{g}} \dots \Sigma_{n_d,k}^{\hat{g}})$. The probability distribution in (2.6) can be written as,

$$\mathcal{N}\Big(\mu_k^{\hat{g}}, \Sigma_k^{\hat{g}}\Big) = \frac{1}{(2\pi)^{n_d/2}\det(\Sigma_k^{\hat{g}})^{1/2}} \exp\left(-\frac{1}{2}(x_k - \mu_k^{\hat{g}})^T {\Sigma_k^{\hat{g}}}^{-1}(x_k - \mu_k^{\hat{g}})\right) \tag{2.7}$$

and is axis-aligned by Assumption 2.1.2.

Remark 2.1.2. *In this thesis the training inputs (x_k, u_k) are assumed to be known with certainty (and hence so is d_k by way of computing it from trajectory information as is formalized in Section 3.3). In practical applications, this will almost never be the case since there will always be uncertainty in blocks like localization in the autonomy stack.*

In light of Remark 2.1.2, GP training that deals with the training inputs themselves being uncertain *distributions* (e.g., as might be produced by state estimators such as Kalman filter) is referred to as "distributional" GP regression. [89] proposes the addition to a corrective variance term to the posterior predictions to address this issue. Alternate approaches involve using distance metrics as covariance functions for input distributions [9]. Popular libraries such as [53] support the implementation of such approaches.

2.2 Introduction to MPC

The aim with MPC is to control a system optimally over a finite horizon or a short period of time into the future. Deterministic continuous-time nominal system dynamics of the form $\dot{x}(t) = f(x(t), u(t))$ yield an infinite-dimensional control problem over arbitrary horizons due to the continuous nature of the optimization variables. The dynamics can be discretized using a step-size Δt that yields a tractable NLP of the form,

$$\min_{x,u} \; \|(x_N - x_N^r)\|_Q + \sum_{k=0}^{N-1} \|(x_k - x_k^r)\|_Q + \|(u_k - u_k^r)\|_R \tag{2.8a}$$

$$\text{subject to} \quad x_{k+1} = x_k + \Delta t f(x_t, u_t), \quad k \in \{0, 1, \ldots, N-1\} \tag{2.8b}$$

$$x_k \in X, \quad u_k \in U, \quad k \in \{0, 1, \ldots, N-1\} \tag{2.8c}$$

$$x_0 = x_{\text{init}} \tag{2.8d}$$

(2.8a) is representative of the cost of deviating from some trajectory x_k^r, u_k^r that is desired to be tracked. In the case of stabilization, $x_k^r = x_s \; \forall \; k \in \{0, 1, \ldots, N-1\}$. In this thesis, reference trajectories will be generated by approaches that do not account for residual dynamics. As such, $u_k^r = 0 \; \forall \; k \in \{0, 1, \ldots, N-1\}$ to provide the online controller with the flexibility to adjust the input as is necessary. The provided input trajectory can however be used for warmstarting [139] the online optimization. (2.8b) uses an Euler discretization scheme to approximate the continuous non-linear dynamics and enforce constraints regarding the system evolution over the horizon. Going forward, the discretized nominal dynamics will be denoted using $x_{k+1} = f(x_k, u_k)$. (2.8c) defines the state and input constraint sets X, U respectively. In this thesis, these sets will be assumed to be box constraints as in most practical applications. Finally, (2.8d) imposes an initial value constraint on the optimization.

Figure 2.2 depicts a typical MPC paradigm where at each timestep t, an *open-loop* optimization over an N step predictive horizon into the future is performed. Then after applying only the first optimal control action, closed-loop state feedback is introduced which yields the following benefits,

- Allows finding better solutions in a receding-horizon manner due to the approximation of an infinite horizon optimal control problem with a finite horizon one for the purposes of computational tractability.
- Allows deviations from the predicted trajectory, due to uncertain dynamics and noise encountered in the real-world, to be accounted for.

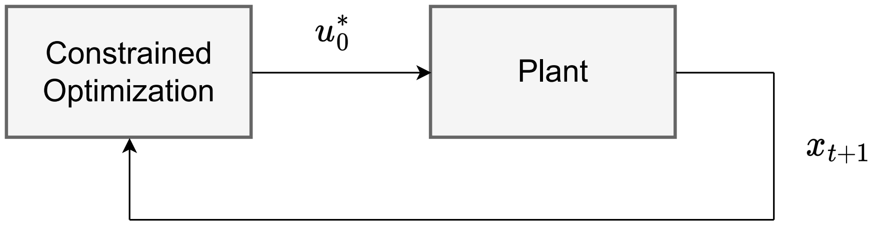


Figure 2.2: A closed-loop MPC controller where the first optimal control action is applied to the plant. The measured next state is passed back to the controller, re-parameterizing the initial value constraint and shifting the reference trajectory after which the optimization is re-solved.

2.2.1 Tractability of Gaussian Process-based MPC

The need for higher fidelity models and data-driven residuals results from the low fidelity models considered when working with real systems not being accurate enough for predictions to remain valid over a longer horizon, especially when operating the system at its limits. However, data-driven approaches trained using libraries external to the optimization toolbox of choice are essentially black boxes to the optimizer.

Optimization-based solvers work to try to make sure no constraints are violated while simultaneously trying to minimize cost. As shown in the diagram, the initialization point violates the equality constraint and the solver can compute a gradient to push this point in the direction of the normal to the line hence reducing constraint violation. Since our data-driven models appear in the dynamics constraints, it is crucial to be able to compute these gradients efficiently. There are 2 factors that affect the computation of these gradients particularly when using auto-differentiation frameworks (e.g., [7]).

- **Dataset size.** When using ML models that rely on inference conditioned on datasets for their prediction step, the computational cost of the prediction scales with the number of training points in the dataset. GPs are no exception to this rule. In this thesis, it is assumed that the dataset size is relatively small ($\sim$100s of points) although there are several sparse GP approximations [140, 41, 123] that can readily be incorporated into the proposed frameworks if this is not the case.

- **Software implementation.** Most, if not all, auto-differentiation engines benefit in speed from constraints implemented within their native API/framework. Sometimes provisions are made to allow for black-box constraint terms to be included in the optimization (e.g., [7] callbacks), although this can significantly affect computation

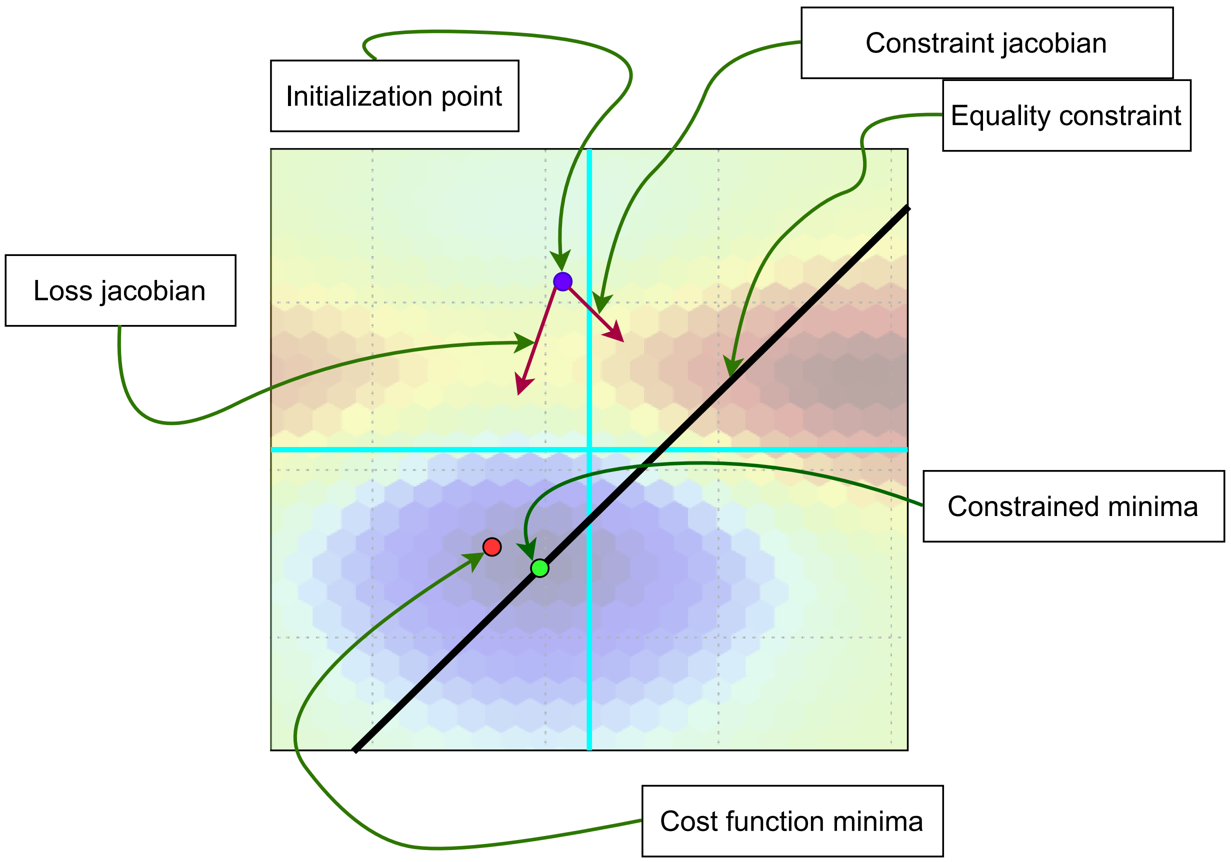


Figure 2.3: A visual demonstration of the workings of gradient-based optimizers

time. The prediction step for GPs (2.5) is straightforward to implement in any optimization framework of choice since it has a simple linear algebraic form. Hence, computing gradients is quicker in comparison to other bayesian methods where this prediction step might not be as easy to encode within the optimization framework.

2.2.2 Stochastic Chance-Constrained MPC

Polytopic sets are often used when expressing constraint sets on the state of the system. One special type of polytope is that of a box constraint. This involves constraints of the form,

$$x_{i,lb} \leq x_i \leq x_{i,ub} \; \forall \; i \; \in \; \{0, 1, \dots, n_x\} \tag{2.9}$$

where n_x denotes the dimension of the state. This can be formulated into a linear constraint of the form,

$$Hx \leq b \tag{2.10a}$$

$$H = \begin{bmatrix} -I \\ I \end{bmatrix} \tag{2.10b}$$

$$b = \begin{bmatrix} -x_{lb} \\ x_{ub} \end{bmatrix} \tag{2.10c}$$

Polytopes such as (2.10) can be written as a conjunction of half-space constraints,

$$\{x \mid Hx \leq b\} \equiv \{x \mid \wedge_{j=1}^{N_x} h_j^T x \leq b_j\} \tag{2.11}$$

where j is used to index the j^{th} row and entry of H and b respectively. In the case of the box constraint, it is clear from (2.10b) that $N_x = 2n_x$.

Assumption 2.2.1. *The constraint sets on state and input are of the form described in* (2.10).

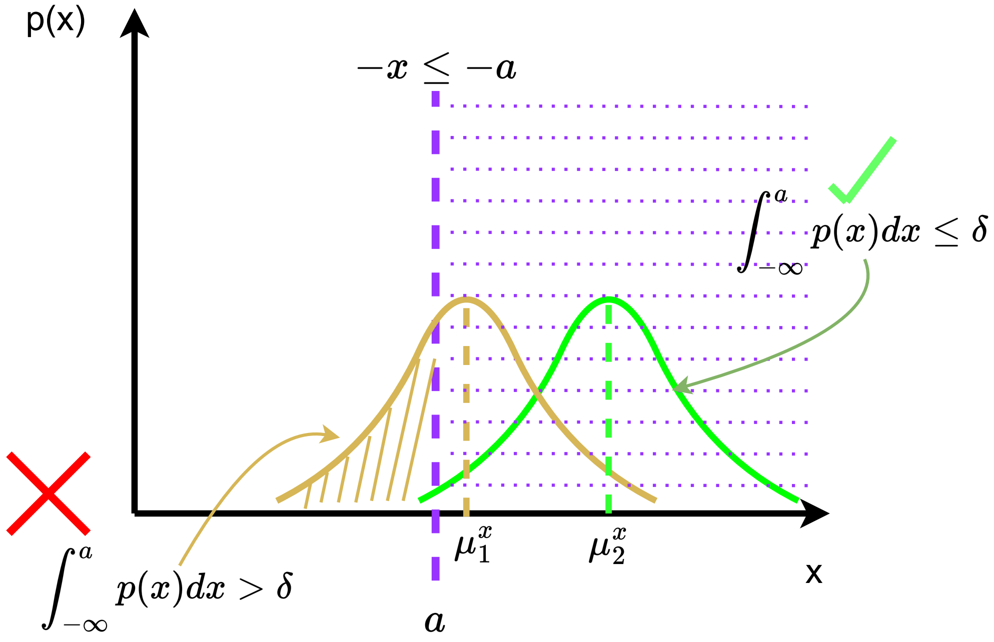
When dealing with stochastic systems, there is uncertainty involved at every timestep of a trajectory prediction across a horizon. Using dynamics models involving terms that represent stochasticity using probability distributions with infinite support (as is done in GPs), the hard constraints defined by (2.10) can no longer be enforced with absolute certainty. Instead, the optimization now deals with probabilistic joint constraint of the form [92],

$$P(\wedge_{j=1}^{N_x}(h_j^T x_k \leq b_j)) \geq 1 - \Delta_k \implies P(\vee_{j=1}^{N_x}(h_j^T x_k > b_j)) \leq p_x \; \forall \; k \; \in \; \{0, 1, \dots, N\} \tag{2.12}$$

where Δ_k represents the joint probability with which the constraints can be violated at each timestep and p_x is the corresponding probability of satisfaction. Using Boole's inequality to provide a union bound on the RHS of (2.12) allows us to re-write it as a set of individual constraints,

$$P(\wedge_{j=1}^{N_x}(h_j^T x_k \leq b_j)) \geq 1 - \Delta_k \iff \wedge_{j=1}^{N_x} \left(P(h_j^T x_k \leq b_j) \geq 1 - \delta_{jk}\right) \tag{2.13a}$$

$$\sum_{j=1}^{N_x} \delta_{jk} = \Delta_k \tag{2.13b}$$



p(x)
$-x \leq -a$
$\int_{-\infty}^{a} p(x)dx \leq \delta$
$\int_{-\infty}^{a} p(x)dx > \delta$
μ_1^x
μ_2^x
x
a

Figure 2.4: Inspired by [23]. A visual demonstration of how the mean of the state affects constraint satisfaction for a given variance term. For both distributions, the means μ_1^x, μ_2^x satisfy the constraint $-x \leq -a (x \geq a)$. For the distribution in green, the left-sided tail does not exceed the specified violation probability while the distribution in yellow does. Thus the state mean plays a big role in constraint satisfaction.

Note: The distribution of probability across the horizon has not been shown here but it can be done in exactly the same manner as has been done for the polytopes at a specific timestep. This modifies (2.13b) to $\sum_{k=1}^{N} \sum_{j=1}^{N_x} \delta_{jk} = \sum_{k=1}^{N} \Delta_k = \Delta$.

Remark 2.2.1. *The simplest uniform risk allocation strategy is utilized in this thesis obtained by defining $\Delta_k = \Delta/N \ \forall \ k \ \in \ \{0, 1, \dots, N\}$, $\delta_{jk} = \Delta_k/N_x \ \forall \ j \ \in \ \{0, 1, \dots, N_x\}$. However, this can lead to over-conservative constraints [92] and there are better approaches to identify how to allocate risk across the individual chance constraints [98].*

Because of the probabilities involved in the constraints in (2.13), it is not possible to represent these in a standard optimization framework. When dealing with arbitrary distributions, several approaches involve using the Cantelli–Chebyshev inequality [88, 92, 103]. However, if $P(h_j^T x_k \leq b_j)$ follows a gaussian distribution, a better (less conservative) deterministic reformulation of (2.13) is obtained by providing constraints on the mean.

Given a multivariate normal distribution over the state vector at timestep k, $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$, an error distribution, e_k^x, can be defined as follows,

$$e_k^x \sim \mathcal{N}(\mathbf{0}, \Sigma_k^x) \tag{2.14a}$$

$$x_k = \mu_k^x + e_k^x \tag{2.14b}$$

Consider now a simple half-space constraint given by $h^T x \leq b$. The distribution of the error in the direction of this constraint is $h^T e \sim \mathcal{N}(\mathbf{0}, h^T \Sigma^x h)$. The chance-constrained version of this half-space constraint when x is a random variable of the form specified in (2.14) can be written using the standard inverse Gaussian Cumulative Distribution Function (CDF) $\phi^{-1}(p_{jk})$ rescaled for the distribution under consideration as follows [92, 79],

$$\mathcal{Z}(\Sigma^x) = \left\{ \mu^x \mid h^T \mu_x \leq b - \phi^{-1}(p) \sqrt{h^T \Sigma_k^x h} \right\} \tag{2.15}$$

where $p = 1 - \delta$ is the probability of satisfaction of the individual chance constraint and $\mathcal{Z}$ denotes a shrunk version of X that depends on the variance of the random variable x. Figure 2.4 shows why it is intuitive that the resulting constraint is placed on the mean.

This can now be generalized to the joint chance-constrained version of the problem to which Boole's inequality has been applied (2.13). Letting $\mathbf{1}_j$ denote the indicator function for the j^{th} dimension of the state and setting $h_j = \mathbf{1}_j$, it follows that,

$$h_j^T e_k^x \sim \mathcal{N}(\mathbf{0}, h_j^T \Sigma_k^x h_j) \tag{2.16a}$$

$$\mathbf{1}_j^T e_k^x \sim \mathcal{N}(\mathbf{0}, \mathbf{1}_j^T \Sigma_k^x \mathbf{1}_j) \tag{2.16b}$$

$$e_{jk}^x \sim \mathcal{N}(\mathbf{0}, [\Sigma_k^x]_{j,j}) \tag{2.16c}$$

where $[\Sigma_k^x]_{j,j}$ denotes the j^{th} diagonal element of Σ_k^x.

It is clear from Figure 2.4 that the inverse CDF is a one-sided condition due to the integration up from $-\infty$. It is desirable to have a symmetric condition on the error which can be generated using $-\mathbf{1}_j$ giving,

$$\mathcal{E}_k^x = \left\{ e_k^x \mid \left(e_{jk}^x \leq \phi^{-1}(p_{jk}) \sqrt{[\Sigma_k^x]_{j,j}} \right) \wedge \left(-e_{jk}^x \leq \phi^{-1}(p_{jk}) \sqrt{[\Sigma_k^x]_{j,j}} \right) \; \forall \; j \in \{0, 1, \ldots n_x\} \right\} \tag{2.17a}$$

$$= \{ e_k^x \mid |e_{jk}^x| \leq \phi^{-1}(p_{jk}) \sqrt{[\Sigma_k^x]_{j,j}} \; \forall \; j \in \{0, 1, \ldots n_x\} \} \tag{2.17b}$$

It is apparent that combining the indicator functions and defining the set for all $j \in \{0, 1, \ldots n_x\}$ yields a box-constrained error set of the form in (2.10). Since there are $2n_x$ constraints, $\delta_{jk} = \frac{1-p_x}{2n_x}$ and hence $p_{jk} = 1 - \delta_{jk}$ where p_x is the probability with which the constraints should be satisfied at each timestep.

Chapter 3

Designing a Hybrid CC-MPC Controller

3.1 Introduction

In this chapter, I consider a CC-MPC control problem for systems subject to “piecewise” (multi-modal) non-linear residual dynamics. The piecewise nature involves the use of discrete variables to allow for switching between the different modes of the dynamics. Conventional approaches neglect the consideration of such discrete variables that involve a subset of the joint state-input vector. For example, GPs are often trained in practice while neglecting the workspace variables from the input space. This is necessary to prevent overfitting and help with data efficiency. However, if the dynamics are piecewise non-linear over the workspace, then the inclusion of discrete variables, as a separate function of the workspace variables, allows us to take these switching dynamics into account.

Hence, I consider the case where the joint state-input space is partitioned into different “regions”, whose locations are known apriori, each of which has a particular mode of the residual active within it. I design a hybrid GP model and demonstrate its benefits over a single GP model when approximating piece-wise functions. I then develop a hybrid controller that leverages these learnt dynamics exhibiting better closed-loop performance and safety when compared to an existing baseline MPC approach. Since these MINLPs are NP-hard to solve in general, I propose an algorithm for a hierarchical planner-controller architecture to reduce the problem complexity, while retaining the benefits, and improve the application potential to systems that demand low controller sampling times.

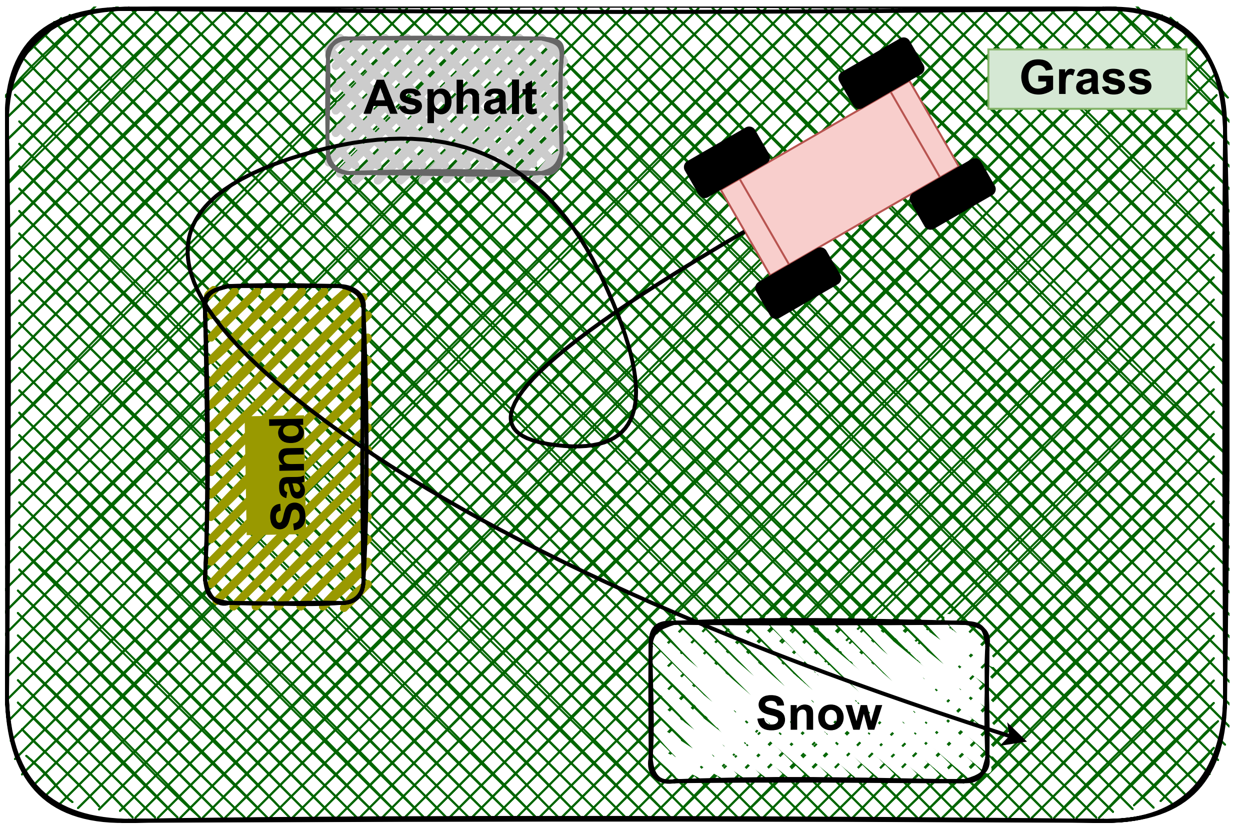


Figure 3.1: A wheeled robot driving over different terrains. The contact dynamics of the tyres will differ across regions and also within the regions, making it difficult for a single model to accurately capture the robot dynamics across the entire workspace.

3.1.1 Motivating Example

Here, I provide an example to practically motivate the problem considered which will be referred to during the rest of this chapter. Consider a wheeled robot operating in a workspace with regions consisting of different terrains as shown in Figure 3.1. Given a nominal model for the robot dynamics, one can learn the unmodeled dynamics or a *residual dynamics* model from data collected over several runs. Due to the robot (tyre) dynamics involved here, that depend on hard to measure environmental factors such as the coefficient of friction, not all regions possess the same true underlying dynamics.

For example, terrains like asphalt, with higher friction, may have lower in-region variation as opposed to slippery surfaces, like ice for the same value of state variables like velocity.

The relevance of the piecewise dynamics becomes apparent when considering a reference path to be tracked containing points belonging to different regions (terrains in this case) across an open-loop horizon. This necessitates the use of a *hybrid* residual dynamics model, to allow for model characteristics to change over regions of the state-input space of a dynamics model, e.g., over different regions in the workspace of this example.

3.2 Problem Statement

Notation

Continuous system variables. I consider a system with state $x_k \in X \subset \mathbb{R}^n$, input $u_k \in U \subseteq \mathbb{R}^m$ at timestep k. Vectors in the joint state-input space are defined as $z_k = [x_k^T, u_k^T]^T \in Z = X \times U$. Z is partitioned into a set of "regions", $R_{\text{set}} = \{R_i \subseteq Z, i \in \{1, \dots, \mathcal{R}\}\}$.

True system dynamics. $f(x_k, u_k) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is used to denote the known, low-fidelity model of the dynamics, i.e., the *nominal* model. $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents the unmodeled (residual) dynamics, which is a piecewise varying function over regions R_{set}. Each mode of this function is denoted by $g_r \; \forall \, r \in \{1, \dots, \mathcal{R}\}$.

Learnt system dynamics. $\hat{g}$ is used to denote the learnt approximation to the true piecewise residual dynamics. The construction formalized here assumes access to full state information. However, for ease, the projection of z_k onto the subspace, G_{in}, of inputs to the GP model is defined as $y^g : Z \to G_{in}, y^g(z_k) = B_{g,in} z_k = y_k^g$. Here, $G_{in} \subseteq \mathbb{R}^{n_g}$ where n_g is known since Assumption 2.1.1 holds.

Discrete (switching) variables. The discrete variables, that allow switching between the modes of the piecewise function g, are indexed by region, r, and timestep, k, and denoted as $\delta_{r,k}$. In a similar fashion to y_k^g, I use $y^\delta : \mathbb{R}^{m+n} \to \mathbb{R}^{n_\delta}, y^\delta(z_k) = B_\delta z_k = y_k^\delta \in \Delta_{in}$ to denote the projection of z_k onto the subspace, Δ_{in}, that defines the regions in R_{set} and controls the assignments to $\delta_{r,k}$. In the context of Example 3.1.1, Δ_{in} is defined by the workspace variables (x, y) with $n_\delta = 2$. ■

The following assumptions and remarks are made regarding R_{set} and the elements that constitute it.

Remark 3.2.1. *The variables in Δ_{in} allows R_{set} to be dependent on either the system (e.g., velocity variables) or the environment (e.g., position variables) or both. In Example 3.1.1,* *it is expected that the locations of the terrains would change with the environment the system is operating in causing them to be environment-dependent.*

Assumption 3.2.1. *The elements in R_{set} are polytopes and the polytopic constraints (of the form $Hy^\delta \leq b$) that define them are known. These regions are also assumed to be time-invariant.*

$Hy^\delta \leq b$ can equivalently be replaced by $Hz \leq b$ (with slight abuse of notation as the matrices are no longer of the same dimension but this is down to zero-padding) since the truth value of the inequality is independent of the assignments to variables in $Z \setminus \Delta_{in}$. In light of Remark 3.2.1, this assumption implies that when dealing with regions defined either in part or completely by workspace variables, the region locations specific to a particular environment are known in advance.

Assumption 3.2.2. $\cup_i R_i = \Delta_{in}$ *and* $R_i \cap R_j = \emptyset \, \forall \, R_i, \, R_j \in R_{set}, \, i \neq j$*, i.e., the regions partition* Δ_{in}

The true discrete-time dynamics of the system can be formulated as,

$$x_{k+1} = f(x_k, u_k) + g(x_k, u_k) + w_k \tag{3.1a}$$

$$\text{where, } g(x_k, u_k) = \sum_{r=1}^{R} \delta_{r,k} g_r(x_k, u_k) \tag{3.1b}$$

$$\text{and, } \delta_{r,k} = \begin{cases} 1, & \text{if} \{x_k, u_k\} \in R_r \\ 0, & \text{otherwise} \end{cases} \tag{3.1c}$$

Here, w_k is the process noise which is considered to be of the form,

$$w_{r,k} \sim \mathcal{N}(\mathbf{0}, \Sigma_{n,r}) \tag{3.2a}$$

$$w_k = \sum_r \delta_{r,k} w_{r,k} \in \mathbb{R}^n \tag{3.2b}$$

where $\Sigma_{n,r}$ characterizes the noise covariance in region r.

Assumption 3.2.3. $\Sigma_{n,r}$ *is a diagonal matrix implying that noise samples are spatially uncorrelated across all dimensions of the residual.*

The following constrained optimal control problem is considered, defined over the system in (3.1).

Problem 1. *Given the system in (3.1) with the unmodeled piecewise dynamics (which is stochastic due to the assumed probabilistic process noise defined in (3.2)), design a controller to minimize the control objective* $J = \sum_k \mathbb{E}_{w_k}(||x_k - x_k^{ref}||_Q^2 + ||u_k||_R^2)$ *while satisfying constraints* $Pr(x_k \in X) \geq p_x$, $Pr(u_k \in U) \geq p_u \, \forall k \geq 0$. *Here, X and U are convex sets, Q and R are positive semi-definite and positive definite matrices respectively, and p_x and p_u are user-defined probabilities of constraint satisfaction at each timestep on the state and input respectively.*

Dataset construction using apriori collected samples. Since the piecewise unmodeled dynamics g are unknown, I make the following assumption in order to turn the problem into one for which a hybrid residual dynamics model can be learnt.

Assumption 3.2.4. *Access is provided to state and input data* $\mathcal{D} = \{x_k, u_k\}_{k=0}^{D}$ *generated by multiple runs of the system (3.1). It is assumed this data is not subject to measurement noise. The data is* rich *in the sense that it covers all regions R_i that partition Δ_{in} and the samples specific to each region are uniform randomly distributed over the domain defined by G_{in}.*

Given these data samples, the mismatch, d_k, can be defined between the actual system (3.1) and a nominal model where $\bar{x}_{k+1} = f(x_k, u_k)$. This information can be encapsulated in an augmented dataset $\tilde{\mathcal{D}}$ defined as,

$$\begin{aligned}
\bar{x}_{k+1} &= f(x_k, u_k) \\
x_{k+1} &= f(x_k, u_k) + \sum_r \delta_{r,k} g_r(x_k, u_k) + w_k \\
d_k &= x_{k+1} - \bar{x}_{k+1} \\
&= \sum_r \delta_{r,k} g_r(x_k, u_k) + w_k \qquad (3.3) \\
\tilde{\mathcal{D}} &= \{z_k, d_k\}_{k=0}^{D} \qquad (3.4)
\end{aligned}$$

Note. For generality here, d_k is demonstrated as the difference between the measured and nominal-predicted state vector, x_{k+1} and $\bar{x}_{k+1}$ respectively. However, in the implementation, it is easier to work with $d_k = y_k^g - \bar{y}_k^g$ where $\bar{y}_k^g = B_{g,in}\bar{x}_{k+1}$ as a result of Assumption 2.1.1.

The computation of d_k, is facilitated by

- The absence of measurement noise (Assumption 3.2.4) and the provided nominal model f, allowing us to generate $\bar{x}_{k+1}$ (the next state as predicted by the nominal model) from x_k, u_k.

- Access to the true measured next state x_{k+1} since $\tilde{\mathcal{D}}$ is generated *after* a trajectory has been executed which allows us to leverage "future" information for the sample at the k^{th} timestep.

The dataset $\tilde{\mathcal{D}}$ can now be used to learn the hybrid residual model as will be further elucidated in Section 3.3.

Note: When dealing with noise models possessing infinite support (e.g., (3.2a)), learning probabilistic models for the residuals that capture this behaviour and embedding this into the controller yields a chance-constrained optimization that *approximately* solves problem 1.

Problem 2. *Given the nominal system dynamics f and the dataset $\tilde{\mathcal{D}}$ to learn a model for the unmodeled piecewise dynamics g, develop a controller for the system* (3.1) *that minimizes the expected cost $J = \sum_k \mathbb{E}_{w_k}(||x_k - x_k^{ref}||_Q^2 + ||u_k||_R^2)$ subject to the approximated chance-constraints:*

$$Pr(x_k \in X) \geq p_x \quad \forall \;\; k \geq 0 \tag{3.5a}$$

$$Pr(u_k \in U) \geq p_u \quad \forall \;\; k \geq 0 \tag{3.5b}$$

3.3 Learning Hybrid GP residual models

In this section, an approach to train a hybrid GP model is presented. The dataset, $\tilde{\mathcal{D}}$, defined in (3.4), is first partitioned into $\{\tilde{\mathcal{D}}_1 \dots \tilde{\mathcal{D}}_{\mathcal{R}}\}$ such that

$$\tilde{\mathcal{D}}_r = \{(z_k, d_k) | (z_k, d_k) \in \tilde{\mathcal{D}} \;\&\; y_k^{\delta} \in R_r\} \; \forall \, r \in \{0, \dots, \mathcal{R}\} \tag{3.6}$$

Using Assumption 3.2.2, it follows that $\tilde{\mathcal{D}}_i \cap \tilde{\mathcal{D}}_j = \emptyset \, \forall \, i, \, j \in \{1, \dots, R\}, \, i \neq j$.

Remark 3.3.1. *For simplicity here, $\tilde{D}_r$ is formalized to be constructed using samples collected in a particular region R_r. When dealing with an environment-dependent Δ_{in} (Remark 3.2.1), this procedure can be generalized to any arbitrary environment instead of just the environment the system is operating in.*

In the context of Example 3.1.1, Remark 3.3.1 essentially says that having access to samples collected on snow in some environment can be used to train a mode of the hybrid GP model that can then be leveraged in any arbitrary environment where snow is present.

Combining the fact the $g_r(x_k, u_k)$ is deterministic $\forall r$ and w_r is zero-mean i.i.d noise with variance $\Sigma_{n,r}$, (3.6) can be alternatively written as,

$$\tilde{\mathcal{D}}_r = \{(z_k, d_k)|(z_k, d_k) \in \tilde{\mathcal{D}} \ \& \ d_k \sim \mathcal{N}(g_r(x_k, u_k), \Sigma_{n,r})\}$$

GPs $(\hat{g}_1 \dots \hat{g}_{\mathcal{R}}) = \hat{g}_{set}$ can now be trained to try to approximate the piecewise residual dynamics defined by $(g_1 \dots g_{\mathcal{R}})$. This allows prediction of d_k as a function of z_k by conditioning on datasets $(\tilde{\mathcal{D}}_1 \dots \tilde{\mathcal{D}}_{\mathcal{R}})$ respectively as outlined in (2.5). The output produced by each of these GPs is denoted as,

$$\hat{g}_r(x_k, u_k) \sim \mathcal{N}\left(\mu^{\hat{g}_r}(\tilde{D}_r, z_k), \Sigma^{\hat{g}_r}(\tilde{D}_r, z_k)\right) \tag{3.7a}$$

$$= \mathcal{N}\left(\mu_k^{\hat{g}_r}, \Sigma_k^{\hat{g}_r}\right) \tag{3.7b}$$

Thus the proposed model is obtained to be,

$$\hat{g}(x_k, u_k) \sim \mathcal{N}\left(\mu_k^{\hat{g}}, \Sigma_k^{\hat{g}}\right) \tag{3.8a}$$

$$\text{where, } \mu_k^{\hat{g}} = \sum_r \delta_{r,k}\mu_k^{\hat{g}_r}, \Sigma_k^{\hat{g}} = \sum_r \delta_{r,k}\Sigma_k^{\hat{g}_r} \tag{3.8b}$$

$\mu_k^{\hat{g}}, \Sigma_k^{\hat{g}}$ represent the residual mean and covariance respectively of the hybrid GP-based residual model and $\delta_{r,k}$ is as defined in (3.1c). Assumption 2.1.2 holds for each $\hat{g}_r \in \hat{g}_{set}$.

Note. The true noise value sampled from the environment is purely dependent on the region and hence $\delta_{r,k}$ dependent on y_k^{δ} (3.2). In general, the (epistemic) uncertainty in the learnt GP dynamics due to the possible lack of availability of samples also contributes to $\Sigma_k^{\hat{g}}$. This causes the uncertainty term in the hybrid residual dynamics to be dependent on y_k^{g} as well rather than just y_k^{δ}.

(3.8) is in contrast to the single GP residual model defined as,

$$\hat{g}_b(x_k, u_k) \sim \mathcal{N}\left(\mu_k^{\hat{g}_b}, \Sigma_k^{\hat{g}_b}\right) \tag{3.9}$$

with the non-linear functions $\mu^{\hat{g}_b}, \Sigma^{\hat{g}_b}$ being dependent on the full dataset $\tilde{D}$.

Remark 3.3.2. *When it comes to switching between region-specific modes, $\hat{g}_r$, note that $\sum_r \delta_{r,k} = 1 \, \forall \, k$ as a direct consequence of Assumption 3.2.2. Hence, the system can only be within one region at any given timestep. Due to this, even though each individual GP itself has infinite support (or more practically, support across the entire bounded space*

G_{in}), the discrete variables defined in the hybrid setup allow the model to switch between different modes based on the region the system is currently in. In other words, the influence of a region-specific GP mode on the residual dynamics ends at the boundaries of its corresponding region.

In summary, the true state dynamics equation (3.1) is approximated as below,

$$x_{k+1} = f(x_k, u_k) + \hat{g}(x_k, u_k) \tag{3.10}$$

with $\hat{g}(x_k, u_k)$ as specified by (3.8).

Assumption 3.3.1. *f and $\hat{g}_r$ are at least 1-time differentiable.*

This is not a particularly limiting assumption since the SE kernel (2.3) used is infinitely differentiable, as are other kernels [47, 110].

3.4 Demonstrating the benefits of hybrid GP models

Firstly, the benefits provided by the hybrid model over a single GP model are demonstrated by looking at several test cases.

The significance of the problem introduced in Section 3.1.1 is visually elucidated using several examples of residual dynamics. The capabilities of the hybrid model, proposed in Section 3.3, to abate the issues faced by a single GP model is demonstrated. Section 3.4.1 provides an in-depth quantitative and qualitative analysis of how a single GP might suffer from smooth approximations to sharp transitions between dynamics from one region to another. Section 3.4.2 and 3.4.3 provide qualitative analyses for additional possible residual cases that might be encountered in practice relating to issues with frequency variation in the dynamics across regions and when the GP input space overlaps across multiple modes respectively.

3.4.1 Effects of discontinuities in residuals.

Though GPs are quite expressive, discontinuities in residual dynamics at region boundaries are approximated by transition regions of non-infinite slope. As a result, for the case where a single GP is used to model residuals of the form (3.1b), a large jump in the residual dynamics between two neighbouring regions can cause a larger region over which the predicted mean dynamics remain highly inaccurate. Tracking values within this transition region can contribute to inaccuracy in predicting of the residual term over the horizon. The switching dynamics (3.8b) ingrained in the hybrid GP model proposed solves this issue relating to transition regions.

As a simple example, a residual that is piece-wise with 2 regions each having independent noise covariances can be considered, as seen in Fig. 3.2 (a). In this specific case, $\Delta_{in} = G_{in} = X$ for simplicity. So to generate the dataset $\tilde{D}$, x_k is sampled uniform randomly from $X = R_1 \cup R_2$ with R_1, R_2 as defined below.

$$R_1 = \{-2 \leq x \leq -0.5\} \; ; \; \Sigma_{n,1} = (0.04)^2$$
$$R_2 = \{-0.5 < x \leq 2\} \; ; \; \Sigma_{n,1} = (0.01)^2$$

$g(x_k, u_k)$ is then obtained using (3.1b) and w_k is sampled from (3.2a) to generate d_k (3.3) given nominal dynamics $x_{k+1} = 0.5x_k + 0.75u_k$.

With this example, it is intended to show how using a hybrid GP model solves the transition region issue while learning a more accurate covariance function across the domain X. For an arbitrary subset $S \subseteq X$ the errors in learnt mean and covariance function from the hybrid model are defined as,

$$e_\mu(S) = \int_{x \in S} |g(x) - \mu^{\hat{g}}(x)| dx \tag{3.12a}$$

$$e_\sigma(S) = \int_{x \in S} |\sqrt{\Sigma_n} - \sqrt{\Sigma^{\hat{g}}(x)}| dx \tag{3.12b}$$

where $\mu^{\hat{g}}(x), \Sigma^{\hat{g}}(x)$ are as in (3.8b) and $\Sigma_n = \sum_r \delta_{r,k} \Sigma_{n,r}$. The same can be used to compute errors for the single GP case using $\mu^{\hat{g}_b}(x), \Sigma^{\hat{g}_b}(x)$ as in (3.9). $e^h_\mu(S), e^h_\sigma(S)$ will be used to refer to errors in the mean and covariance functions for the hybrid model and similarly $e^b_\mu(S), e^b_\sigma(S)$ is used for the single GP model.

Method	$e_\mu(X)$	$e_\mu(R_1)$	$e_\mu(R_2)$	$e_\sigma(X)$	$e_\sigma(R_1)$	$e_\sigma(R_2)$
Single GP	0.731	0.304	0.426	1.49	0.55	0.934
Hybrid GP	**0.345**	**0.193**	**0.1519**	**0.151**	**0.1**	**0.05**

Table 3.1: Comparison of absolute errors in the mean (e_μ) and covariance (e_σ) functions for the hybrid and single GP models over the domains X, R_1, R_2. Errors computed over 30 runs using 400 samples to train the model and 120 values across X for verification to generate the error terms (3.12).

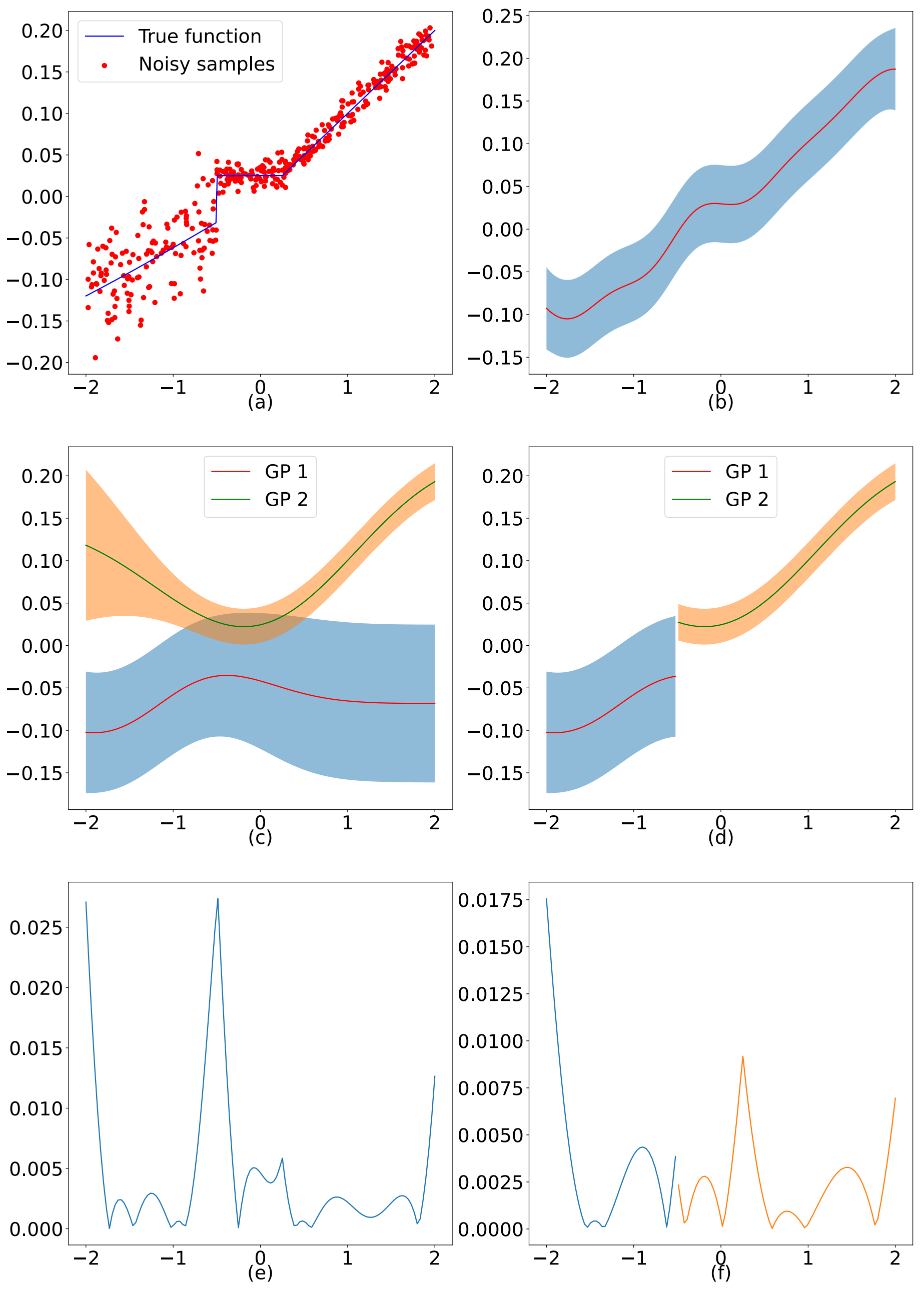


Figure 3.2: Plots of the residual magnitude (y-axis) vs. the single state variable input to the GP (x-axis) (a) Plot of the true underlying piece-wise residual dynamics and the samples generated from it to be used to train the GP. (b) Plot of the single GP and the learnt covariance function. (c) Plot of the independent region-specific trained GPs visualized over the entire state space. (d) Plot of hybrid piece-wise trained GP with mean and variance computed as defined in (3.8b). (e) Mean function absolute error for single GP approach over state domain (f) Mean function absolute error for hybrid GP approach over state domain

Addressing transition regions and learnt mean functions. Fig. 3.2 (b) clearly illustrates the transition region issue for the single GP case along with Fig. 3.2 (f) that shows a spike in mean function error around the region boundary $x = 0.5$.

Fig. 3.2 (c) shows the mean and covariance functions of the individual GPs $\hat{g}_r$ across the entire state space X. However, the switching dynamics 3.8b that result from the introduced discrete variables allow the hybrid model to only consider the residual as depicted in Fig. 3.2 (d) thus solving the transition region issue as further illustrated by the hybrid GP mean function error plot in Fig. 3.2 (f). Moreover, the values in Table 3.1 show a significant reduction in modeling error across the entire state domain, X, as well as the individual regions, R_1, R_2 for the hybrid GP model as compared to the single GP.

Addressing errors in estimated covariance function. When dealing with noise models of the form (3.2), the single GP approach learns a weighted average of the parameters in different regions. Operating within a region that has a higher true noise covariance than the one learnt by the GP covariance function leads to shrinking of constraint sets (as described in Section 2.2.2) in a way that affords a lower degree of conservatism than would be desired to meet the specified satisfaction probabilities from (3.5) and vice versa.

Remark 3.4.1. *There exist GPs capable of modelling "heteroscedastic" noise i.e., noise that is a function of the input [137, 84, 21]. Their use is not considered here since they are not only harder to train but also because they are unnecessary when dealing with a noise model of the form* (3.2) *which varies only with region and not continuously across the entirety of the GP input space.*

The hybrid model learns separate noise models (and/or parameters) for every region to address this problem resulting in it significantly outperforming the single GP in estimating the residual dynamics (3.3) (and uncertainty) as seen in Table 3.1.

Note. The "kink" or discontinuity in the function causes the single GP to learn a very short lengthscale [47]. Thus, if the data is not dense, the uncertainty grows much faster for the single GP as opposed to the hybrid model for the same dataset as seen in Figure 3.3 which deals with a residual similar to Figure 3.2 but with a slightly exacerbated discontinuity between the piecewise modes.

3.4.2 Addressing issues with learnt lengthscales.

The second case considered is when the residual dynamics models in each region have varying frequency components. A simple example of this is a sum of sinusoids with varying

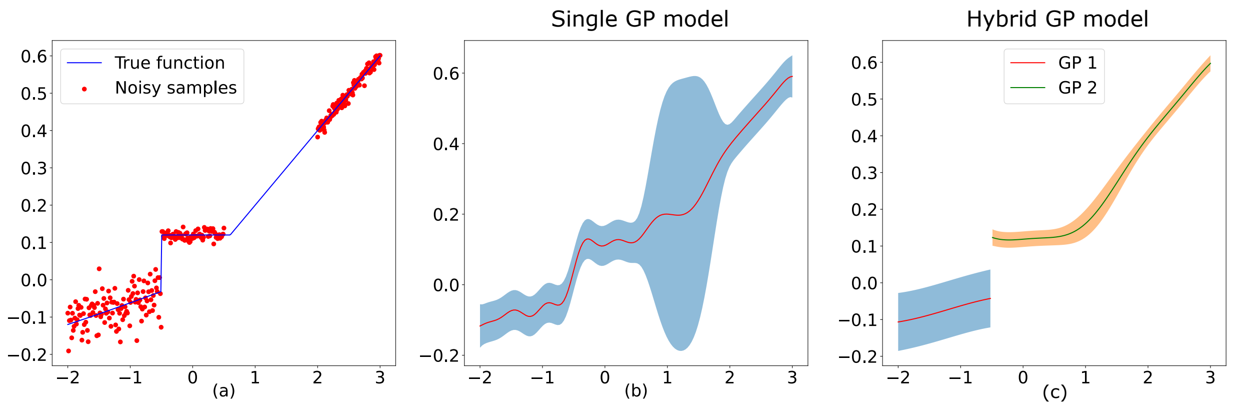


Figure 3.3: (a) Shows a piecewise residual with a significant discontinuity between its modes. (b) Shows the output after training a single GP model. The short learnt lengthscale introduces noise into the mean function and causes uncertainty to rise sharply in areas of low data density. (c) Shows the output from a hybrid GP model that is able to extrapolate the high lengthscale piece-wise mode well in the absence of data.

frequencies. As depicted previously in Figure 2.1, the learnt lengthscale determines how fast the function varies and is hence related to the frequency components in the underlying model it is trying to approximate. Figure 3.4 (c) shows the case where the single GP hyperparameter optimization converges to the smaller lengthscale determined by region 2. It is clear that the GP starts locally overfitting to noisy samples in region 1 due to this small learnt lengthscale as evidenced by the high frequency artefacts that seems to appear on top of the true underlying function in the learnt model over this region. Alternatively, Figure 3.4 (b) shows the case where the lengthscale convergence is dominated by the samples in region 2. Here, the larger learnt lengthscale causes the single GP model to believe the samples collected in region 2 are extremely noisy and "filters" out the high frequency components here. As a result, the noise parameter learnt by hyperparameter optimization is very large and generalizes to region 1 as well even though the samples collected in that region clearly have very little noise.

Remark 3.4.2. *It could be possible to use the rational quadratic [47, 110] kernel which learns a weighted combination of lengthscales (albeit constant across the entire input space) to partially address this issue for the single GP. Alternatively, kernels with long and short learnt lengthscales could be combined to ensure that no region is devoid of their true lengthscale but this still means there is inevitably high/low frequency components injected across the entire space even where they might be undesirable.*

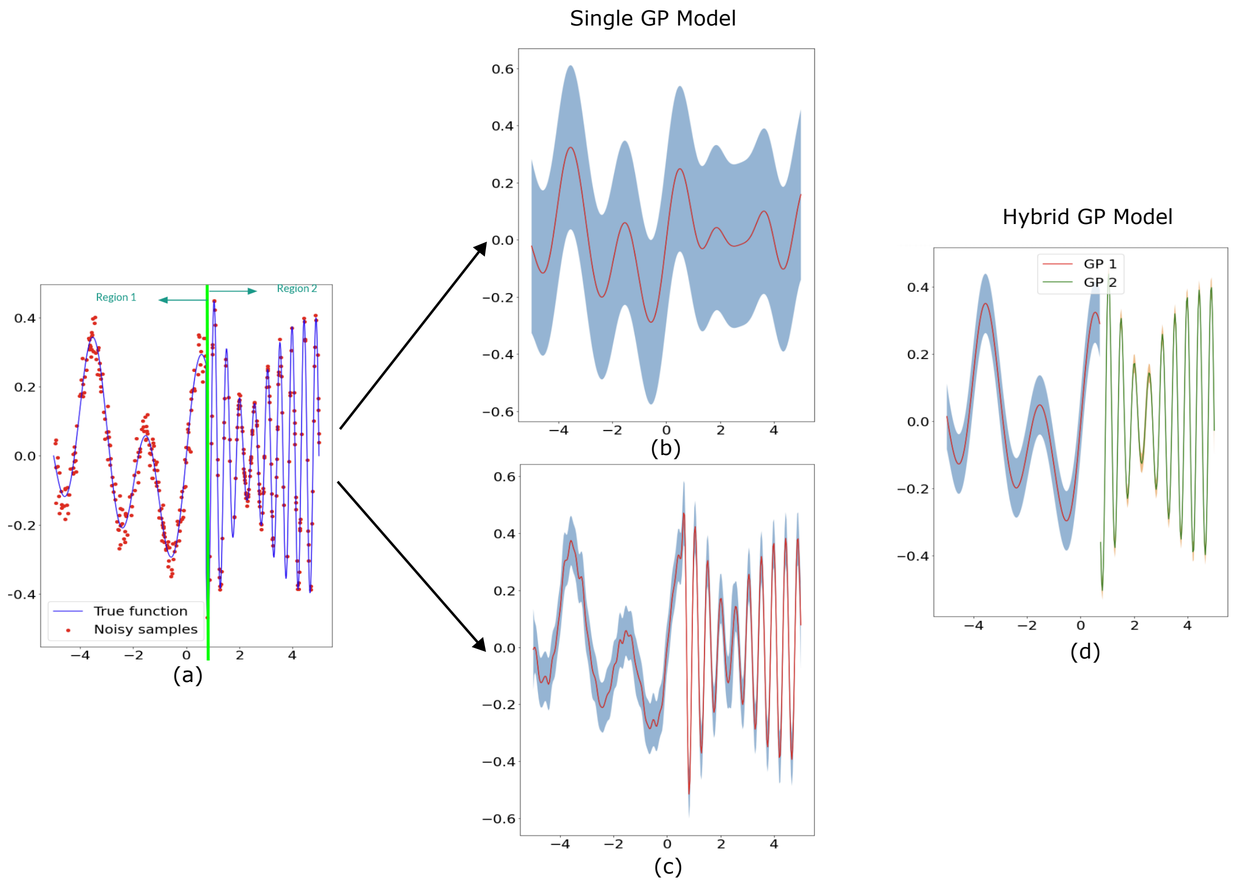
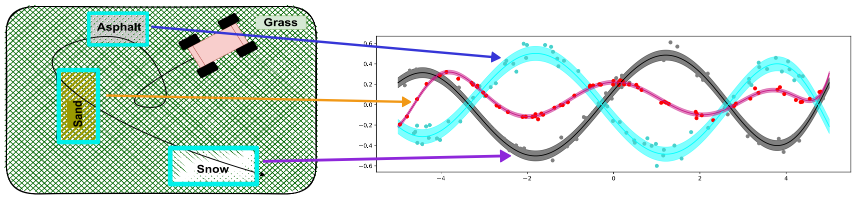
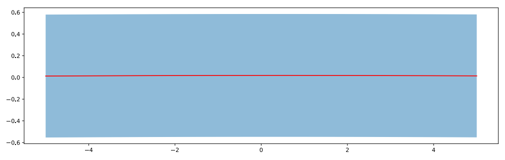


Figure 3.4: (a) True sinusoidal residual underlying dynamics with a high frequency sinusoid in region 2. (b) A learnt single GP model where hyperparameter optimization has aligned with the parameters in region 1. (c) A learnt single GP model where hyperparameter optimization has aligned with the parameters in region 2. (d) Learnt hybrid GP model with 2 modes that remedies the issues of the single GP models.

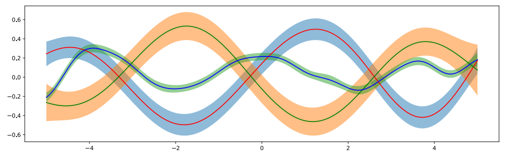
In contrast, the hybrid GP model proposed addresses the possible issues that arise when using the single GP model as seen in Figure 3.4 (d).



(a)



(b)



(c)

Figure 3.5: Illustrative example. (a) Visual mapping of different terrains across the workspace to the corresponding modes of the residual model. (b) A single GP model which learns a very noisy function due to the low correlation between the different modes across the input space. (c) The hybrid GP model that accurately learns the piecewise residual dynamics with an overlapping input space.

3.4.3 Residual dynamics with overlapping input spaces.

The two cases discussed so far have considered examples where the modes of the true piecewise dynamics have non-overlapping input domains i.e., $\Delta_{in} = G_{in}$. As indicated in Example 3.1.1, this is not the case the general.

When the residual dynamics have overlapping input domains, the single GP model performs significantly worse as has been demonstrated in Figure 3.5 (b). It is clear that the posterior mean of the learnt single GP across the entire input space is still constantly zero. If one were to use this in control, it would essentially resemble a nominal MPC controller with added conservativeness due to the large learnt noise parameter.

In contrast, the hybrid GP model in Figure 3.5 (c) does not suffer from the same issue.

3.5 Designing an SMPC Controller for a model with hybrid piece-wise dynamics

3.5.1 Control Policy.

Since searching over the space of all possible control laws is generally infeasible, one usually defines a form of control law to optimize over. Here an "indirect" feedback control law [66] is considered which is defined as,

$$u_k = \pi_k(x_k) = K(\mu_k^x - x_k) + \mu_k^u \tag{3.13}$$

The term "indirect" comes from the fact that feedback of the state mean μ_k^x is not directly considered as is conventional [92] but rather consider the deviation of the true state from the mean. [64] compares the two approaches and shows how the indirect feedback law prevents over-conservatism that might result from using the direct feedback law.

Remark 3.5.1. *[62] shows how the controller performs well in a practical application to miniature race cars even in the absence of the feedback term in the control law (i.e., K=0).*

Remark 3.5.1 could be relevant in the case where the discretized uncertainty contributed at every timestep is not too large thus preventing a rapid compounding growth of uncertainty over the horizon.

$K : \mathbb{R}^n \to \mathbb{R}^m$ is chosen to be the stabilizing feedback matrix after solving the LQR problem (using Q, R defined as in Problem 1) that reflects the cost of deviation from a reference trajectory and actuation effort respectively.

The above control structure defines μ_k^u as the deterministic control variable while the actual control input u_k itself is stochastic due to its dependence on x_k.

3.5.2 Propagating residual dynamics across a horizon.

I now incorporate the proposed hybrid GP, approximating the piecewise residual dynamics, into the controller design. For reasons that will become clearer over the rest of this section, the distribution over joint state-input-residual vectors is *approximated* by a multivariate

Gaussian distribution at every timestep.

$$\begin{bmatrix} x_k^T & u_k^T & \hat{g}_k^T \end{bmatrix}^T \sim \mathcal{N}(\mu_k, \Sigma_k) \tag{3.14a}$$

$$\text{where, } \mu_k = \begin{bmatrix} (\mu_k^x)^T & (\mu_k^u)^T & \left(\mu_k^{\hat{g}}\right)^T \end{bmatrix}^T \tag{3.14b}$$

$$\text{and, } \Sigma_k = \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & \Sigma_k^{x\hat{g}} \\ \Sigma_k^{ux} & \Sigma_k^u & \Sigma_k^{u\hat{g}} \\ \Sigma_k^{\hat{g}x} & \Sigma_k^{\hat{g}u} & \Sigma_k^{\hat{g}} \end{bmatrix} \tag{3.14c}$$

where $\mu_k^{\hat{g}}, \Sigma_k^{\hat{g}}$ are as in (3.8), μ_k^u as in (3.13) with the matrix, Σ_k, being symmetric by definition. The process of obtaining μ_{k+1}^x and the individual terms in (3.14c) is explained below.

Propagating state mean and covariance. Given nominal system dynamics, a first order Taylor series approximation can be used to linearize them around the mean. Using the approximated dynamics (3.10) allows for the propagation of the joint Gaussian distribution, Σ_k, through the linearized function while maintaining that the next state (output) remain a Gaussian as well [61] characterized by the mean and covariance shown below.

$$\mu_{k+1}^x(\mu_k) = f(\mu_k^x, \mu_k^u) + B_g \mu_k^{\hat{g}} \tag{3.15a}$$

$$\Sigma_{k+1}^x = [\nabla f(\mu_k^x, \mu_k^u) \; B_g]\, \Sigma_k [\nabla f(\mu_k^x, \mu_k^u) \; B_g]^T \tag{3.15b}$$

Note. The matrix B_g is used to limit the residual terms to specific dimensions of the state vector and can be useful to neglect learning unnecessary residuals when certain components of the dynamics are well known. Here, $B_g : \mathbb{R}^{n_g} \to \mathbb{R}^n, \mu_k^{\hat{g}} \in \mathbb{R}^{n_g}, \Sigma_k^{\hat{g}} \in \mathbb{R}^{n_g \times n_g}$. Formulations up to this point can be assumed to have used $B_g = I_{n\times n}$ with $\mu_k^{\hat{g}} \in \mathbb{R}^n$.

Assumption 3.5.1. *The initial state is known perfectly, i.e.,* $\Sigma_0^x = \mathbf{0}_{n\times n}$ *implying* $\Sigma_0^u = \mathbf{0}_{m\times m}$ *since* $\mu_0^x = x_0$ *in* (3.13).

$\Sigma_k^x \;\forall\; k \geq 1$ can then be obtained using (3.15b).

Constructing joint state-input-residual covariance matrix: From (3.13) it is clear that Σ_k^u, Σ_k^{xu} are merely linear transformations of Σ_k^x computed as $\Sigma_k^u = K\Sigma_k^x K^T$, $\Sigma_i^{xu} = \Sigma_i^x K^T$. In Section 2.1 the predicted output distribution for a deterministic input was discussed. However, when dealing with a look-ahead prediction procedure (as introduced in Section 2.2), the input to the GP after the first time step is drawn from a multivariate Gaussian distribution with covariance $B_g \Sigma_0^{\hat{g}} B_g^T$ (using (3.14c) (3.15b) and Assumption 3.5.1). In general, since the mean function of a GP is nonlinear in general, propagating a Gaussian distribution through it results in a non-Gaussian output distribution that is intractable to evaluate analytically. While there are sampling based methods capable of approximating this distribution [76][97], in this thesis I focus on methods that model the output distributions at each timestep k by a *approximate* multivariate Gaussians [73][55].

For simplicity, I make use of the mean equivalence approximation for uncertainty propagation which assigns $\Sigma_k^{x\hat{g}} := \mathbf{0}_{n\times n_g}$, $\Sigma_k^{u\hat{g}} := \mathbf{0}_{m\times n_g}$, $\Sigma_k^{\hat{g}} := \Sigma^{\hat{g}}(\mu_k^x, \mu_k^u)$. This has the benefit of lower computational complexity at the cost of compounding errors in uncertainty over longer horizons. Other approximations such as Taylor approximation and exact moment-matching can be found in [73, 52, 55].

3.5.3 Shrunk chance-constrained sets.

The formulation in Section 3.5.2 has yielded equations for Σ_k^x at every step k over the open-loop horizon. For a general polytopic constraint set X, the following theorem from [61] can be used for obtaining shrunk constraint sets based on the box-constrained error sets (2.17) derived in Section 2.2.2.

Theorem 3.5.1. *[61] Given an error set of the form (2.17), the shrunk constraint set for the state mean, μ_k^x, can now be obtained using the Pontryagin difference [26] as,*

$$\mu_k^x \in \mathcal{Z} = X \ominus \mathcal{E}^x(\Sigma_k^x) \tag{3.16a}$$

$$\mathcal{Z}(p_x, \Sigma_k^x) = \{\mu_k^x \mid H\mu_k^x \leq \tilde{b}(\Sigma_k^x)\} \tag{3.16b}$$

with $\tilde{b}(\Sigma_k^x) = b - |H|\phi^{-1}(\bar{p})\sqrt{diag(\Sigma_k^x)}$, $\bar{p} = \frac{1-p_x}{2n_x}$ and ϕ^{-1} being the inverse CDF described in Section 2.2.2. $diag(\cdot)$ yields the vector of diagonal elements and square-root is taken element-wise.

For an indirect feedback control law with $\Sigma_k^u = K\Sigma_k^x K^T$, shrunk constraint sets for the input can be similarly derived,

$$\mu_k^u \in \mathcal{V} = U \ominus \mathcal{E}^u(\Sigma_k^u) \tag{3.17}$$

3.5.4 Big-M formulation for the hybrid residual model.

Given a point belonging to a particular region the optimizer must be provided with a way of selecting the corresponding mode of the hybrid GP for the dynamics update step. This

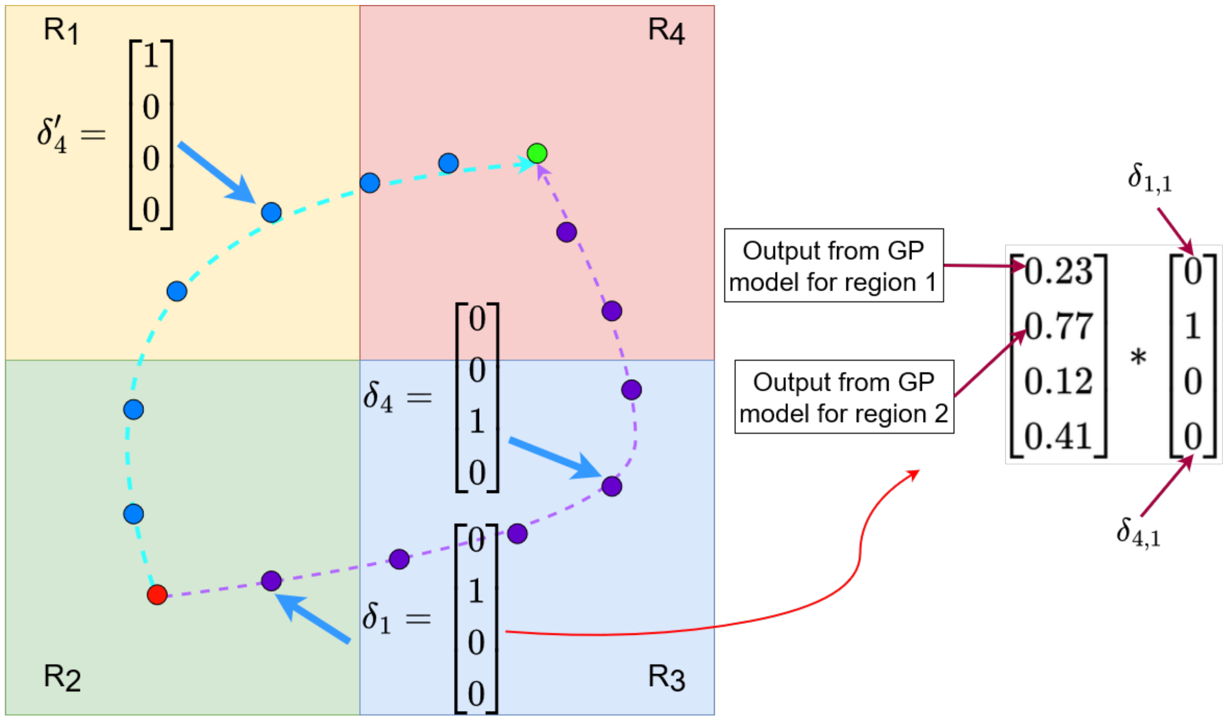

R1
R4
R2
R3
$\delta'_4 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$
$\delta_4 = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$
$\delta_1 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$
Output from GP model for region 1
Output from GP model for region 2
$\begin{bmatrix}0.23\\0.77\\0.12\\0.41\end{bmatrix} * \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$
$\delta_{1,1}$
$\delta_{4,1}$

Figure 3.6: Optimizing over the space of trajectories to get from the start position (red) to the destination (green) leads to different assignments to the discrete δ variables at each timestep. These assignments can be element-wise multiplied with the vector of residual outputs from the hybrid GP model. As a simple example, consider the third timestep of the purple path and assume it belongs to the 2nd region. In this case, since region 2 is the active mode, the residual term from that region is selected for the dynamics equation.

cannot be done apriori since optimizing over the space of trajectories yields trajectories that can have differing modes active at different timesteps.

To handle for this, a set of discrete variables is introduced which evaluate to 1 when a point is within the region and 0 otherwise as exemplified in Figure 3.6. The residual terms are computed in parallel for all modes of the hybrid model. However, only the term corresponding to the active (as determined by the assignment to the discrete variables) mode is selected.

Dealing with polytopic non-overlapping regions (Assumptions 3.2.1 and 3.2.2), the corresponding constraint that replicates (3.1c) can be defined as,

$$H_r z_k \leq b + \vec{M}(1 - \delta_{r,k}) \tag{3.18}$$

where $\vec{M} \gg 0$ and is of the same dimension as b_r. Since $\delta_{z,k} \in \{0, 1\}$ this yields,

$$(\delta_{r,k} = 1) \implies H_r z_k \leq b_r \implies z_k \in R_r \tag{3.19a}$$

$$(\delta_{r,k} = 0) \implies H_r z_k \leq b_r + \vec{M} \tag{3.19b}$$

where (3.19b) is always true so long as $\vec{M}$ is set to be large enough to encompass Z and hence reduces to $\delta_{r,k} \implies z_k \in Z$. Since $\mathcal{Z} \times \mathcal{V}$ from (3.16a) is already contained within this set, (3.19b) is essentially a redundant implication that indirectly says $\exists r \in \{1 \ldots \mathcal{R}\}$ s.t. $\delta_{r,k} = 1 \, \forall \, k$. In order to constrain $\delta_{r,k} = 1$ only when $z_k \in R_r$, I enforce the additional constraint,

$$\sum_r \delta_{r,k} = 1 \; \forall \; k \geq 0 \tag{3.20}$$

Thus, combining (3.18) and (3.20) a re-formulation of (3.1c) can be obtained. **Note.** The dimension of the array of discrete (binary) variables involved in the formulation is indicated below,

$$\delta \in \{0, 1\}^{R \times N} \tag{3.21}$$

Remark 3.5.2. *The inclusion of these discrete variables at each timestep allows for switching between modes across the horizon. This is in contrast to [96] which learns a weighted convex combination of a library of GP models to construct a residual model that remains constant over the horizon. Whether this is beneficial in practice or not is left for future work.*

3.5.5 Cost Function.

Dealing with the stochastic system in (3.1), a quadratic cost function computed as an expectation over states and inputs [1] can be considered. This is defined as,

$$\begin{aligned}\mathbb{E}(x_k - x_k^r, u_k - u_k^r) =& \|\mu_k^x - x_k^{ref}\|_Q^2 + \|\mu_k^u - u_k^r\|_R^2 \\ &+ \mathrm{trace}(Q\Sigma_k^x) + \mathrm{trace}(R\Sigma_k^u)\end{aligned} \tag{3.22}$$

utilizing Q, R as in 3.5.1. The terminal cost is defined as,

$$\mathbb{E}(x_N - x_N^r) = \|\mu_N^x - x_N^r\|_P^2 + \mathrm{trace}(P\Sigma_k^x) \tag{3.23}$$

where P is the solution to the Discrete Algebraic Riccati Equation (DARE).

3.5.6 Hybrid MPC Formulation

By bringing together the constructions of the previous sections, the following hybrid MPC is proposed to solve problem 2.

$$\begin{aligned}
&\min_{\mu_k^u,\delta} \mathbb{E}(x_N - x_N^r) + \sum_{k=0}^{N-1} \mathbb{E}(x_k - x_k^r, u_k - u_k^r) \\
&\text{s.t. } \Sigma_0^x = \mathbf{0}_{n\times n}\ ,\ \mu_0^x = x_0, \\
&\quad \delta = \{\delta_{r,k}\}_{r\in 0,\dots,\mathcal{R},k\in 0,\dots,N-1} \in \{0,1\}^{\mathcal{R}\times N} \\
&\quad \forall k = \{0,\dots,N-1\}: && (3.24a) \\
&\quad \mu_k^{\hat{g}} = \sum_r \delta_{r,k}\mu_k^{\hat{g}_r}, && (3.24b) \\
&\quad \Sigma_k^{\hat{g}} = \sum_r \delta_{r,k}\Sigma_k^{\hat{g}_r}, && (3.24c) \\
&\quad \mu_k^u \in \mathcal{V}(\Sigma_k^u), \\
&\quad \forall k = \{0,\dots,N\}: \\
&\quad \mu_{k+1}^x(\mu_k) = f(\mu_k^x,\mu_k^u) + B_g\mu_k^{\hat{g}}, \\
&\quad \Sigma_{k+1}^x = [\nabla f(\mu_k^x,\mu_k^u)\ B_g]\,\Sigma_k[\nabla f(\mu_k^x,\mu_k^u)\ B_g]^T, && (3.24d) \\
&\quad \mu_{k+1}^x \in \mathcal{Z}(\Sigma_{k+1}^x),
\end{aligned}$$

where Σ_k in (3.27a) is obtained as described in Section 3.5.2 and δ is the collection of binary variables (3.21). Note, this is a MINLP since the mean and covariance update equations are non-linear and $\delta_{r,k}$ are binary variables, while the control variables μ_k^u are real valued. At every timestep, the control input μ_0^{u*} is applied as generated by the optimizer. The problem is then re-solved in a receding horizon manner at the next timestep after obtaining a new initial state.

Remark 3.5.3. *For simplicity in this formalization, it has been assumed that each individual region contributes to a "unique" mode of the hybrid model.*

In the context of Example 3.1.1 this means that no two disjoint regions across the workspace share the same underlying terrain. It is however, straightforward to extend the proposed controller to settings where this is not the case as follows,

- $\forall R_i, R_j \in R_{set}$ s.t. $i \neq j$, if the regions have the same underlying mode of the residual model, $\hat{g}_t$ then their augmented datasets $\tilde{D}_i, \tilde{D}_j$ are merged. The process can be repeated for each mode of the hybrid model. Let $R_t = \{R_i \mid R_i \in R_{set} \;\&\; \hat{g}_r = \hat{g}_t\}$.
- Let n_t denote the number of regions in the workspace that have a particular mode, t, and N_t denote the total number of modes. Another array of discrete variables $\delta_t \in \{0,1\}^{N_t \times N}$ can be defined. If δ_{in} falls in any of the regions in R_t, then $\delta_t = 1$ i.e., an additional constraint is now defined in the optimization as follows,

$$\delta_k^t = 1 \iff \sum_{i=0}^{n_t} \delta_k^r = 1 \tag{3.25a}$$

- It is the δ^t array that now gets incorporated into the control formulation for selecting the mean and covariances in (3.24b) and (3.24c) respectively.

3.6 Improving computational speed of a hybrid GP-MPC controller.

The results presented in Section 3.7.1 pertaining to the controller proposed in Section 3.5.6 require a large amount of time required to solve the MINLP optimization at each timestep of the closed-loop simulation. This could be useful for systems like chemical plants or HVACs with relatively low controller sampling frequencies. However, MPC controllers are usually expected to operate at a much higher frequency for practical robotic platforms and this problem is now addressed.

3.6.1 Cause of slow MINLP solve times.

MINLPs problem are much harder to solve than NLPs due to the presence of discrete (integer) variables in MINLP problems which causes the search space over which the optimization is performed to suffer from the problem of combinatorial explosion. These algorithms often use techniques such as branch and bound, relaxations and heuristics to address the computational tractability of the problem [25]. In contrast, NLP problems that deal solely with continuous variables rely on comparatively more efficient gradient-based optimization methods to find (sub-)optimal solutions.

The δ array in Section 3.5.6 has $R \times N$ discrete variables as shown in (3.21) where R is the number of "unique" regions (e.g., terrains) involved in the hybrid GP model. For robotic platforms, open-loop MPC over a look-ahead horizon of $\sim$1-2 seconds with a control sampling frequency of 10-20 Hz is common [16, 77]. The larger horizon is particularly necessary in the case where terminal constraints are not included in the optimization [56, 18, 3]. This often corresponds to $N \in [20, 100] \cap \mathcal{Z}$. However, increasing N linearly scales the number of discrete variables and causes the complexity of the resulting optimization to grow out of hand very quickly. It is desirable to retain a high value of N to benefit from the longer look-ahead and the increased prediction accuracy as a result of the available higher fidelity model obtained by incorporating a data-driven dynamics term.

If the δ array is now considered to be parametrized, it is no longer required to search over the space of assignments to the variables in the δ array. While this can affect the optimality of the resulting solution, it significantly improves the computational tractability of the optimization to be solved by the controller.

In order to parametrize the δ array, I propose a hierarchical planner-controller architecture that allows reduction to a simpler NLP problem while retaining the benefits of the hybrid formulation. The proposed approach is first elaborated on in Section 3.6.2 and then considerations that might arise pertaining the planners being used are discussed in Section 3.6.3.

3.6.2 Constructing a parametrized hybrid MPC controller.

Assumption 3.6.1. *Any arbitrary planner can be used so long as it provides assignments to the variables in* $y_k^{\delta} \in \Delta_{in} \ \forall \ k \in \{0, 1, \ldots N_{sim}\}$ *with* $N_{sim} = T/\Delta t$ *with* T *being the simulation time and controller sampling frequency being* $1/\Delta t$.

Given a plan with such assignments, $y_k^{\delta,ref}$, it is now possible to directly evaluate $y_k^{\delta,ref} \in \{y_k^{\delta} \mid H_r y_k^{\delta} \leq b_r\}$ for each region, r, and assign values to the vectors in the δ array 3.21

apriori. Additionally, if assignments to y^g_k are also provided, it is possible to approximately shrink the constraint sets apriori as well. This is due to the fact that $\Sigma^{\hat{g}}_k$ can be computed and propagated through the covariance dynamics equations (3.27a) obtained by linearizing the dynamics about the nominal trajectory.

Notation. Given a tensor, A, I use the indexing shorthand $A[:, i:j, k] = A[:][i:j][k]$, where "[:]" selects all the elements along the first axis and "$[i:j]$" selects elements from indices $i \rightarrow j-1$ (limits inclusive) along the second axis. $len(\cdot)$ is an overloaded operator that returns the number of elements when applied to sets and the number of *columns* when applied to 2-D matrices. $range(N)$ is used to denote the set of whole numbers leading up to (but not including) N i.e., $range(N) = \{0, 1 \dots N-1\}$ ■

Algorithm 1 details how the computation of certain components of the online controller in Section 3.5.6 can be shifted to an offline block and approximated apriori. I now list the considerations embedded within the algorithm,

- Consider an online controller operating over N_{sim} simulation timesteps. Given a particular simulation step, $k_{sim} \in range(N_{sim})$, the reference state trajectory over the N step open-loop horizon at k_{sim} is given by $x^{ref}_k[:, k_{sim} : k_{sim}+N]$ and similarly for the input trajectory. Since the references are required till $k_{sim} = N_{sim} - 1$, $len(x^{ref}) = len(u^{ref}) = N_{sim} + N = N_{total}$ as enforced on Line 3.
- Corresponding to each $y^{g,ref}_{k_{total}}$ with $k_{total} \in range(N_{total})$, each mode in $\hat{g}$ yields a covariance matrix of size $n_d \times n_d$ (as obtained on Line 14). Since there are $\mathcal{R}$ modes in $\hat{g}$, these covariance matrices can be stacked into the tensor, $\Sigma^{\hat{g}}_{hyb}$, after iterating over all regions, $range(R_{set}) = \mathcal{R}$, and all timesteps, $range(N_{total})$. As a result the dimension of $\Sigma^{\hat{g}}_{hyb}$ is $((n_d \times n_d) \times \mathcal{R}) \times N_{total})$, as initialized on Line 9.
- Corresponding to each $y^{\delta,ref}_{k_{total}}$ with $k_{total} \in range(N_{total})$, the polytopic inequalities that define each of the regions in R_{set} can be evaluated to obtain truth assignments to discrete variables δ^{ref} (as done on Lines 15-16). Since there are $\mathcal{R}$ modes in $\hat{g}$, the dimension of δ^{ref} is $\mathcal{R}) \times N_{total}$, as initialized on Line 8.
- Given assignments to δ^{ref} at each timestep $k_{total} \in range(N_{total})$, the covariance of the active mode, r (i.e., $r \quad \text{s.t.} \delta^{ref}_{r,k_{total}} = 1$), can be selected by element-wise multiplication of the region-specific covariance matrices with their corresponding δ^{ref} variable assignment at the given step k_{total} followed by summation as done on Line 17. As a result, since this computation collapses the "region" axis, $\Sigma^{\hat{g}}$ is of dimension $((n_d \times n_d) \times N_{total})$ as initialized on Line 10.

Algorithm 1 Offline block for hybrid Gaussian-Process based MPC.

```
Require: $\hat{g}$: Hybrid GP model
Require: $f$: Nominal dynamics model
Require: $x^{ref}, u^{ref}$: Reference state and input trajectory generated by a planner
Require: $N_{sim}$: Number of simulation timesteps
Require: $B_{g,in}, B_\delta$: As defined in the notation of Section 3.2
Require: $R_{set}$: Set of regions that partition $\Delta_{in}$
Require: $K$: State feedback control matrix from (3.13)
Require: $N$: Horizon length
 1: procedure HGPMPC_PRIOR
 2:   $N_{total} \leftarrow N_{sim} + N$
    // Online controller requires a reference over an $N$ step horizon till $k_{sim} = N_{sim}$.
 3:   assert: $len(x^{ref}) = len(u^{ref}) = N_{total}$
 4:   $z^{ref} \leftarrow [{x^{ref}}^T, {u^{ref}}^T]^T$
 5:   $y^{g,ref} \leftarrow B_{g,in} z^{ref}$
 6:   $y^{\delta,ref} \leftarrow B_\delta z^{ref}$
 7:   $\mathcal{R} \leftarrow \text{len}(R_{set})$
    // Init. quantities with dimensions dictated by assertion on Line 3.
 8:   $\delta^{ref} \leftarrow \mathbf{0}_{\mathcal{R} \times N_{total}}$
 9:   $\Sigma^{\hat{g}}_{hyb} \leftarrow \mathbf{0}_{n_d \times n_d \times \mathcal{R} \times N_{total}}$
10:   $\Sigma^{\hat{g}} \leftarrow \mathbf{0}_{n_d \times n_d \times N_{total}}$
    // Init. state uncertainty matrices over $N$-step horizon at each $k_{sim} \in range(N_{sim})$
11:   $\Sigma^{x,ref} \leftarrow \mathbf{0}_{n_x \times n_x \times N \times N_{sim}}$
12:   foreach $k_{total} \in \text{range}(N_{total})$ do
13:     foreach $j \in \text{range}(\mathcal{R})$ do
14:       $\Sigma^{\hat{g}}_{hyb}[:, :, j, k_{total}] \leftarrow \Sigma^{\hat{g}_j}(y^{g,ref}_{k_{total}})$   ▷ Mode-specific learnt residual covariance.
15:       if $H_j y^{\delta,ref}_{k_{total}} - b_j \leq \mathbf{0}$ then
16:         $\delta^{ref}_{j,k_{total}} \leftarrow 1$   ▷ Otherwise retained at 0.
17:     $\Sigma^{\hat{g}}[:, :, k_{total}] \leftarrow \sum_{j=0}^{\mathcal{R}} \delta^{ref}_{j,k_{total}} \Sigma^{\hat{g}}_{hyb}[:, :, j, k_{total}]$   ▷ Obtain active mode's covariance
18:   foreach $k_{sim} \in \text{range}(N_{sim})$ do
19:     foreach $k \in \text{range}(N)$ do
20:       $\text{idx\_offset} \leftarrow k + k_{sim}$
21:       $\Sigma^{\hat{g}}_k \leftarrow \Sigma^{\hat{g}}[:, :, \text{idx\_offset}]$
22:       $\Sigma^x_k \leftarrow \Sigma^{x,ref}[:, :, k, k_{sim}]$
23:       $\Sigma_k \leftarrow \text{JointCovCompute}(\Sigma^x_k, K, \Sigma^{\hat{g}}_k)$   ▷ As in Section 3.5.2
24:       $\Sigma^{x,ref}[:, :, k+1, k_{sim}] \leftarrow \text{CovDynProp}(f, z^{ref}_{\text{idx\_offset}}, \Sigma_k)$   ▷ As in (3.15b)
25:   return $\delta^{ref}, \Sigma^{x,ref}$
```

- Finally, I aim to generate approximate state uncertainty matrices, $\Sigma^{x,ref}$, to be used for approximate shrinking of the state constraint set, X, for the online controller. At each simulation step, $k_{sim} \in range(N_{sim})$, state covariance matrices of shape $n_x \times n_x$ are required at every step of the open-loop horizon, N, in accordance with 3.16a. As a result, $\Sigma^{x,ref}$ is of dimension $(((n_x \times n_x) \times N) \times N_{sim})$ as initialized on Line 11. $\Sigma^{x,ref}[:,:,k,k_{sim}]$ is utilized for shrinking at open-loop timestep, k, and simulation step, $k_{sim} \in range(N_{sim})$, for the online controller.

- The information contained in the residual covariances $\Sigma^{\hat{g}}[:,:,k_{sim} : k_{sim} + N$ can be used to compute $\Sigma^{x,ref}[:,:,:,k_{sim}]$ as done over Lines 18-24. JointCovCompute on Line 23 is as constructed in Section 3.5.2 and CovDynProp on Line 24 is as specified in (3.15b).

$$
\begin{aligned}
&\min_{u_k} \|\mu_N^x - x_N^{ref}\|_Q + \sum_{k=0}^{N-1} \|\mu_k^x - x_k^{ref}\| + \|u_k\|_R && \text{(3.26a)}\\
&\text{s.t. } \Sigma_0^x = \mathbf{0}_{n\times n}\ ,\ \mu_0^x = x_0, \\
&\quad \forall k = \{0,\dots,N-1\} : && \text{(3.26b)}\\
&\quad \mu_k^{\hat{g}} = \sum_r \delta_{r,k}^{ref} \mu_k^{\hat{g}_r}, && \text{(3.26c)}\\
&\quad u_k \in U, \\
&\quad \forall k = \{0,\dots,N\} : \\
&\quad \mu_{k+1}^x(\mu_k) = f(\mu_k^x, \mu_k^u) + B_g \mu_k^{\hat{g}}, \\
&\quad \mu_{k+1}^x \in \mathcal{Z}^{ref}(\Sigma_{k+1}^{x,ref}),
\end{aligned}
$$

From (3.26a) it is clear that the optimization no longer considers the space of assignments to δ unlike with (3.24) but is now parametrized by a reference δ^{ref} generated by Algorithm 1. Moreover, the optimization cost 3.26a no longer includes the expectation but rather just the deviation of the mean from the reference trajectory. x_k^{ref} is pulled from x^{ref} as passed to Algorithm 1 but shifted to the right closed-loop simulation timestep k_{sim}. u^{ref} can be used for warmstarting [139] u_k^{ref} but is not used as a reference for input tracking since u_k is desired to go to $\mathbf{0}$.

In this NLP-based approach, K is chosen to be $\mathbf{0}$ based on Remark 3.5.1 implying that $u_k = \mu_k^u$ is a deterministic control input and so U is not subject to set shrinking. Thus, the only possible non-zero entries in (3.14c) are Σ_k^x and $\Sigma_k^{\hat{g}}$ ($\Sigma_k^{x\hat{g}} = \mathbf{0}$ by mean equivalence approximation).

The function to compute shrunk sets that constrain the mean at every timestep is denoted by $\mathcal{Z}^{ref}$ and is the same as (3.16a), except computed offline using the $\Sigma^{x,ref}$ tensor returned by Algorithm 1.

Apriori set shrinking and neglecting the expectation from the cost function altogether removes the need to compute any covariance matrices online exemplified by the lack of covariance dynamics equations in (3.26). While this may not seem computationally significant, these symbolic matrices can reduce the sparsity [7] of the optimization problem which makes it harder to solve. As a result, neglecting their inclusion does provide noticeable speed-up.

Remark 3.6.1. *Let R_i be the region active at timestep k (i.e., $\delta^{ref}_{i,k+k_{sim}} = 1$). The controller in* (3.24) *does* not *enforce constraints of the form $\mu^x_k \in R_i$.*

Remark 3.6.1 further showcases a limiting property of the resulting controller i.e., there are no guarantees provided on the true state sampled from the underlying stochastic dynamics being in the region specified by x^{ref}. Moreover, placing constraints on the mean alone would be insufficient since the distribution over states should lie within the region R_i with some probability, p_r (different from p_x, p_u), yielding another chance constraint on the state mean μ^x_k. While these constraint sets might be approximately computed using the reference covariance dynamics encapsulated in $\Sigma^{x,ref}$, it is not something I address in this thesis. This is because it tightly couples the controller to knowing the constraints that define the regions (Assumption 3.2.1) which is a limiting assumption and one I try to alleviate in Chapter 4.

3.6.3 Planning considerations.

Remark 3.6.2. *For the experiments in this thesis, the approach will involve using a simple nominal MPC controller to generate reference trajectories. There are N_W high-level waypoints each of which is tracked for the same number of timesteps i.e., N_{sim}/N_W (although a distance-based weighting might be preferable).*

However, other approaches might be better suited to practical applications depending on the use case.

The granularity of the reference trajectory generated by the planners used becomes crucial to obtaining reasonable parametrizations of δ. If the distance between two successive states in the reference trajectory x^{ref}_k, x^{ref}_{k+1} is much larger than what the control constraints allow the system to reasonably execute, then the controller might significantly deviate

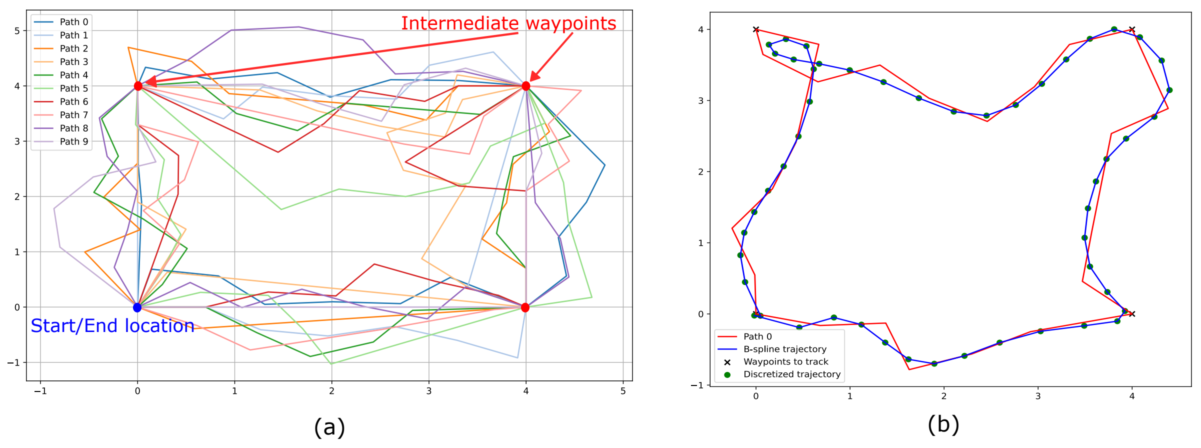


Figure 3.7: (a) An example of RRT outputs over multiple runs trying to find paths between a set of high-level waypoints. (b) An example of a spline-interpolated trajectory generated from the waypoints output by RRT. This trajectory can then be discretized as desired to obtain a reference trajectory for the online controller to track.

from the expected trajectory and the offline computed shrunk sets and the modes used for residual mean prediction across the horizon might no longer be accurate.

A practical approach could be using a planner like RRT* or one of its variants [82, 51]. However, with planners such as these the granularity with which they explore the search space often becomes crucial to obtaining plans quickly. This is usually determined by parameters that control the radius of exploration around existing nodes in the graph. A small radius of exploration could lead to large solve times and so a larger radius is often preferred. Utilizing spline-based methods on top of these high-level plans to generate a smooth trajectory that can further be subsampled (as in Figure 3.7) with the desired granularity would be of importance to generate viable reference trajectories.

In the case, where $G_{in} \cap U = \phi$ and the nominal trajectory generated is not operating at the edge of the input space, if the online controller that takes into account the residual dynamics is able to find a control sequence, u_k, that keeps the trajectory of the state mean close to x_k^{ref}, then the covariance propagation might be more accurate since it is mainly determined by $\Sigma_k^{\hat{g}}$. Inaccuracies in the linearization about the reference input $\frac{\partial f}{\partial u}$ become irrelevant since $K = 0$ i.e., $\Sigma^{u\cdot} = \mathbf{0}$. As a result, the only inaccuracies in the uncertainty propagation would stem from terms involving u in $\frac{\partial f}{\partial x}$.

3.7 Numerical Simulations

3.7.1 Mixed-Integer GP-MPC Controller.

The controller outlined in (3.24) is implemented in Python using Casadi [7] with the BONMIN MINLP solver [25] and IPOPT [136] as the NLP solver. For comparison, a *baseline* GP-based MPC [61] is also implemented that uses a single GP to model the residual dynamics. The GPs were trained using GPyTorch [53]. The proposed methodology is tested on 3 cases, namely a stabilization task and 2 cases to track a set-point at the boundary of the state constraints. For each of the 3 cases, simulations are run from 10 random initial states with 2 runs per initial state (since (3.1) is stochastic), i.e., a total of 20 simulations for both the baseline and proposed method.

Simulation Setup. A planar under-actuated (single input) system is considered for these simulations. For sake of simplicity and ease of visualization, the residual dynamics and uncertainty g are limited to affect the first state variable alone ($B_g = [1\,,\,0]^T$) with the mean function as depicted in Figure 3.8. The residuals and regions are assumed to be purely dependent on the 2 state variables $x_{1,k}$ and $x_{2,k}$ in this case. The nominal dynamics used are of the from $f(x_k, u_k) = Ax_k + Bu_k$ where,

$$A = \begin{bmatrix} 0.85 & 0.2 \\ 0 & 0.95 \end{bmatrix} ; B = \begin{bmatrix} 1.25 \\ 2 \end{bmatrix} \tag{3.27a}$$

For the state constraint set, X, the box constraint -2 $\leq x_{j,k} \leq 2 \quad j \in \{1, 2\} \quad \forall \quad k$ is considered. The process noise considered in (3.2a) takes the form $w_k \sim (\mathbf{0}, \delta_{1,k}\Sigma_{n,1} + \delta_{2,k}\Sigma_{n,2} + \delta_{3,k}\Sigma_{n,3})$ where $\Sigma_{n,1}, \Sigma_{n,3}, \Sigma_{n,3}$ vary with each of the cases as discussed in the following subsections in order to highlight the benefits of the proposed approach across various scenarios. Regions R_1, R_2, R_3 are hyper-rectangular partitions of X with bounds $[-2, -2] \times [-0.5, 2], [-0.5, 0.5] \times [2, 2]$ and $[-0.5, -2] \times [2, 0.5]$ respectively. The desired (minimum) probability of constraint satisfaction is set to be $p_x = p_u = 0.85$ (see (3.5)), unless specified otherwise. The MPC horizon is set to $N = 3$ (a relatively short horizon choice and one that has already been discussed in Section 3.6.1).

To compare the baseline approach with the method I propose, I use the closed-loop (C.L.) cost metric:

$$\text{C.L. cost} = \sum_{k=0}^{T-1} (||x_k - x_k^{ref}||_Q^2 + ||u_k||_R^2) \tag{3.28}$$

Here x_k^{ref} is the pre-defined reference value of the state at time step k. A lower cost implies better control performance since this is the objective to be minimized. Q and R

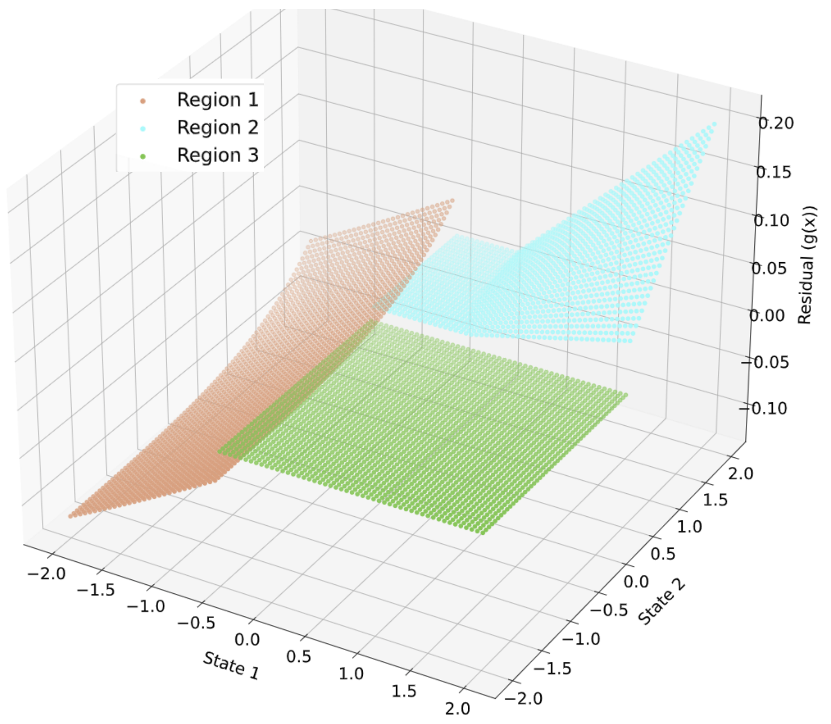


Figure 3.8: Residual dynamics function (g) used for all 3 cases.

are matrices as defined in the MPC formulation. μ_{cl}^{c} is defined as the empirical mean of this cost over multiple runs.

Stabilization around the origin

In this setting, $x_k^{ref} = 0, \forall k$. The input constraint set U is $-0.4 \leq u_k \leq 0.4$. For this example the cost matrices $R = 0.45$, $Q = 0.75 I_{2\times 2}$ are used for the objective of the proposed and baseline MPC controllers. 0-mean process noise (3.2) is considered as defined by $\{\Sigma_{n,1}, \Sigma_{n,2}, \Sigma_{n,3}\} = \{0.075, 0.2, 0.15\}$.

Results. As seen in Figure 3.9, both the baseline and proposed hybrid GP-MPC controller drive the system state close to the origin. The proposed controller keeps the states in a neighborhood of the origin. Over multiple simulation runs, the average C.L. costs for baseline and proposed approach are quite similar as seen in Table 3.2, with the proposed approach showing a lower variance. For this task, both the controllers do not show any constraint violations.

Tracking a setpoint in a region of high process noise at the boundary of the constraint

For this task, I consider the case where $\Sigma_{n,2} \gg \Sigma_{n,1}, \Sigma_{n,3}$. The controller must track

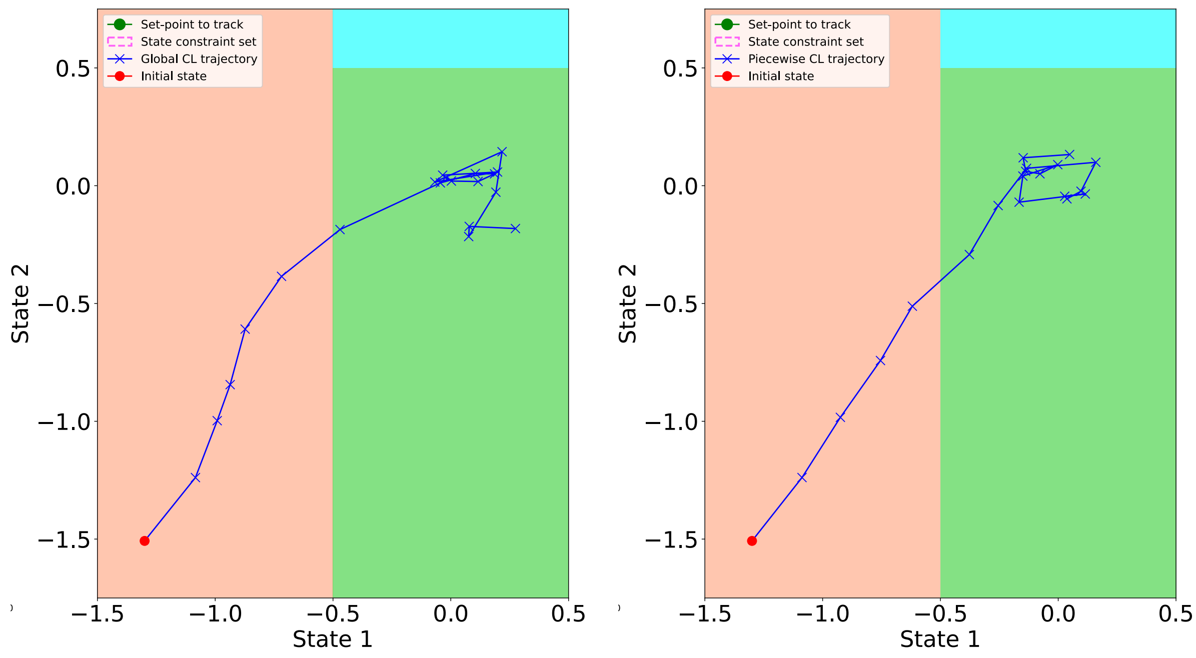


Figure 3.9: Region 1 (peach), region 2 (cyan) and region 3 (green). Baseline GP-MPC (left) and Proposed Hybrid Piece-wise GP-MPC (right) closed loop trajectories for the stabilization task.

Method	CL Cost Mean (μ^c_{cl})	CL Cost Variance
Baseline	11.21	3.85
Proposed Approach	**11.06**	**2.877**

Table 3.2: Mean and variance of the closed loop costs between the baseline and proposed approach over 20 simulation runs for the stabilization task. A lower cost implies better control performance.

a reference point (here, $x^{ref} = [2, 2]^T \in R_2$) at the boundary of X, which will result in trajectories which spend most of their time in R_2. The aim with this setup is to validate the premise that in this case a single GP-based residual model over-approximates shrunk sets in R_2 (as introduced in Section 3.4 and demonstrated by Table 3.1) since it learns $\Sigma^{\hat{g}_b}(\tilde{D}, x_k, u_k) \ll \Sigma_{n,2} \forall (x_k, u_k) \in R_2$. Consequently, this also compromises the control performance of the baseline method. Here, the considered process noise uses the parameters $\{\Sigma_{n,1}, \Sigma_{n,2}, \Sigma_{n,3}\} = \{0.05, 0.4, 0.1\}$ and $(p_x, p_u) = (0.85, 0.85)$ in order to realize this scenario. For a simulation length of 15, this satisfaction probability corresponds to an allowed number of violations of $0.15 * 15 = 2.25$ (time steps) on average. In addition to recording the number of violations the *extent* of violation is also measured which is

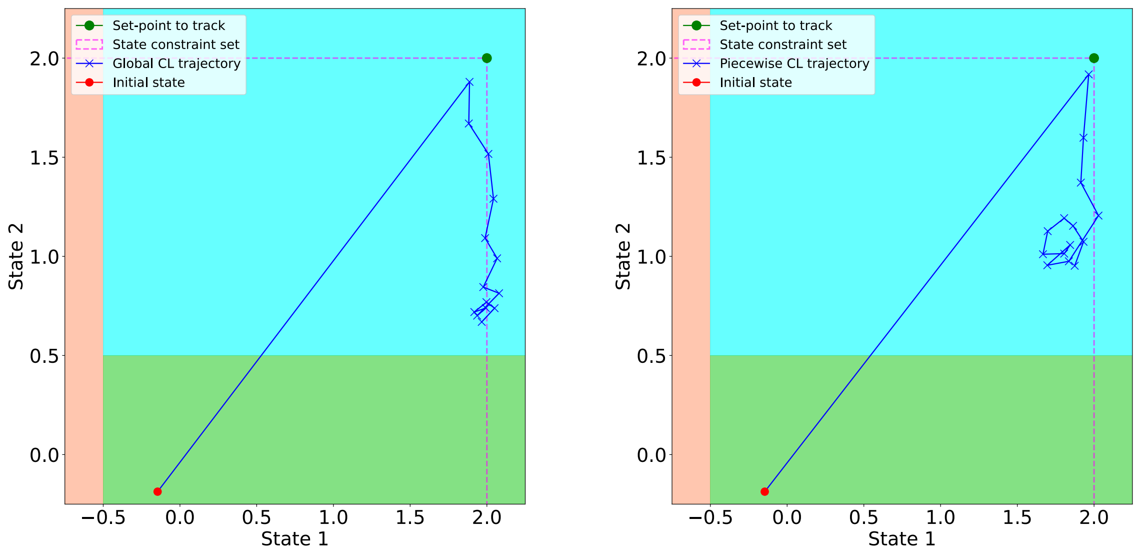


Figure 3.10: Baseline GP-MPC (left) and Proposed Hybrid Piece-wise GP-MPC (right) closed loop trajectories showing constraint violations while tracking a set-point at the state-space boundary.

calculated as,

$$V_{ext} = \sum_{k} \sum_{j} h(|x_{j,k}| - 2) \tag{3.29}$$

where h denotes heaviside step function.

Results. As shown in Table 3.3, the baseline approach achieves a better closed-loop cost here but is in violation of the specified satisfaction probability by a significant amount, indicating that it is not cautious enough and proving the premise. In comparison, the proposed controller proceeds with the right degree of conservatism exhibiting an average violation below the 2.25 limit. It also exhibits a significantly lower extent of constraint violation when compared with the baseline. Figure 3.10 shows one simulation where the baseline controller results in constraint violations at 5 timesteps as opposed to a single violation for the proposed approach.

Tracking a setpoint in a region of low process noise at the boundary of the constraint.

For this task, the case where $\Sigma_{n,2} \ll \Sigma_{n,1}, \Sigma_{n,3}$ is considered. Once again, the set-

Method	$\mu_{\#viol}$	$\sigma_{\#viol}$	$V_{ext}(\mu)$	$V_{ext}(\sigma)$	μ^c_{cl}
Baseline	3.44	1.413	0.219	0.157	**57.99**
Proposed	**1.25**	**1.14**	**0.07**	**0.078**	60

Table 3.3: Averages ($\mu_{\#viol}, V_{ext}(\mu)$) and standard deviation ($\sigma_{\#viol}, V_{ext}(\sigma)$) of the number and extent of state constraint violations with average C.L. cost (μ^c_{cl}) for the baseline and proposed approaches. Computed over 20 simulation runs.

point the controller attempts to track is the same i.e., $x^{ref} = [2,2]^T \in R_2$. Here it is demonstrated that the baseline method under-approximates shrunk sets in R_2 since it learns $\Sigma^{\hat{g}_b}(\tilde{D}, x_k, u_k) \gg \Sigma_{n,2} \forall (x_k, u_k) \in R_2$. For this simulation, the control authority is increased and U is set such that $-0.6 \leq u \leq 0.6$. The cost of control inputs is also lowered by setting $R = 0.05$, and set $Q = 2I_{2\times 2}$. This is to ensure that the controller is not disincentivized from reaching the set point due to the cost of control inputs. The process noise parameters are set as $\{\Sigma_{n,1}, \Sigma_{n,2}, \Sigma_{n,3}\} = \{0.4, 0.05, 0.55\}$ and satisfaction probabilities $(p_x, p_u) = (0.4, 0.4)$.

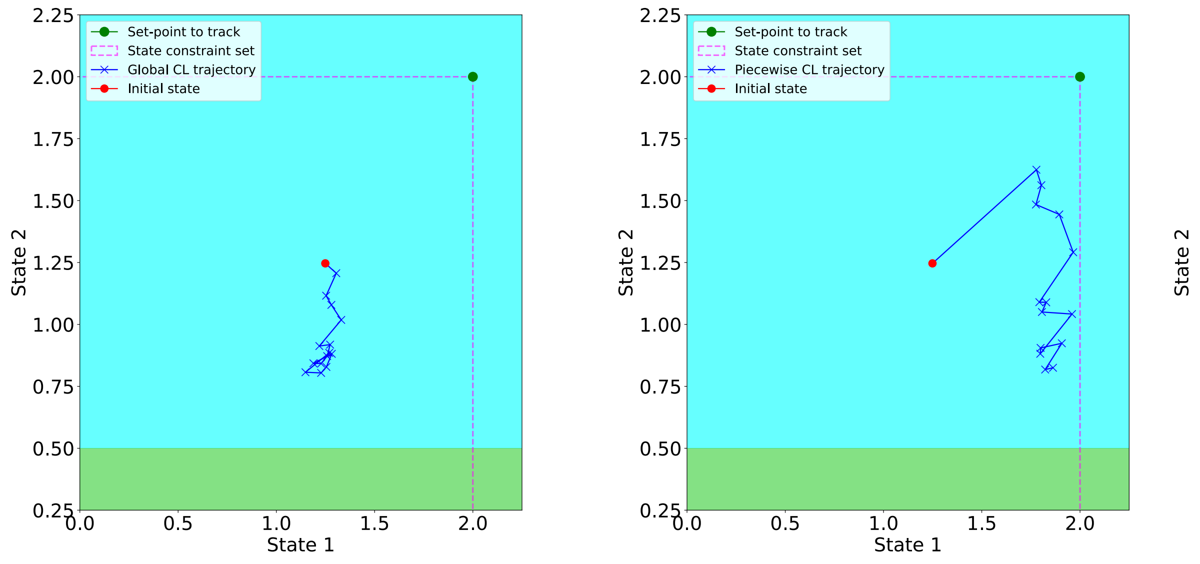


Figure 3.11: Baseline GP-MPC (left) and Proposed Hybrid Piece-wise GP-MPC (right) closed loop trajectories for tracking a set-point at the boundary of the state space.

Results. It can be seen from Figure 3.11 that the baseline method is over-cautious. This significantly affects the closed-loop tracking cost as seen in Table 3.4 causing the proposed method to outperform it by 45% on average. For both controllers here, no constraint

Method	CL Cost Mean	CL Cost Variance
Baseline	53.54	10.66
Proposed Approach	**29.09**	**8.64**

Table 3.4: Mean and variance of the closed loop costs between the baseline and proposed approach over 20 simulation runs for the tracking task. Lower cost represents better control performance.

violations are experienced. For the proposed approach this might be due to the fact that the system is under-actuated (preventing it from tracking the set-point in both dimensions) along with the hyperparameters learning a slightly larger ground-truth noise parameter for R_2 leading to slightly more conservative set shrinking.

3.7.2 Hybrid NLP GP-MPC Controller.

Even for an extremely short horizon of $N = 3$, the controller in (3.24) took an average of 51 seconds to produce a control input. In contrast, the results presented in this section are applied to a more realistic system and significant computational speed-up is demonstrated. The controller outlined in (3.26) is once again implemented in Python using the Casadi [7] framework using IPOPT [136] to solve the resulting NLP.

Simulation Setup.

For this more tractable controller, a 2-D quadrotor dynamics model (as described in Appendix A) is considered for simulation. Here, y_k^δ is chosen to be the workspace variables (x, z) with $y_k^g = v_x$ and the region partitioning visualized in Figure 3.12 (a). The residual affects only the x dynamics i.e., B_g is given by the indicator function $\mathbf{1}_1^T$ since x is the first state variable. The continuous time residual dynamics g and process noise term under consideration w (with discretized version as described in (3.1b)) used in this simulation are as depicted in Figure 3.12 (b). The process noise considered in (3.2a) has the parameters takes the form $\Sigma_{n,R_1} = 0.1, \Sigma_{n,R_2} = 0.075, \Sigma_{n,R_3} = 0.125$. The cost matrices are chosen to be $Q = diag([5, 1, 5, 1, 1, 1])$ (placing importance on tracking the $x - z$ coordinates) and $R = 0.1 I_{2\times2}$.

For the workspace variables, the constraints are assumed to be of the form, $0 \leq x \leq 7$; $0 \leq z \leq 7$. The constraints on the remaining variables in the state vector that define X are system-specific and defined in Appendix A. The MPC horizon is set to $N = 30$ and the discretization time, Δt, used is 20 ms.

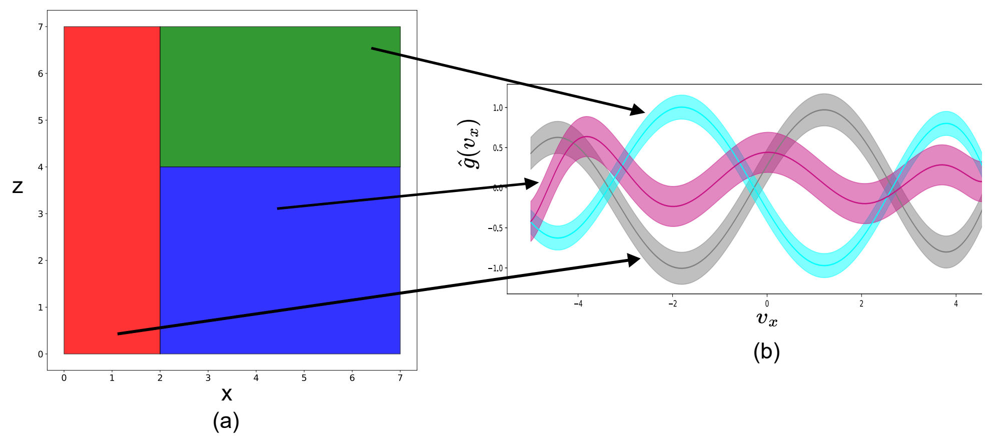


Figure 3.12: A plot depicting the (a) partition of the workspace into 3 regions (b) the residual magnitude and noise for each mode of the piece-wise dynamics and the corresponding region they are active in.

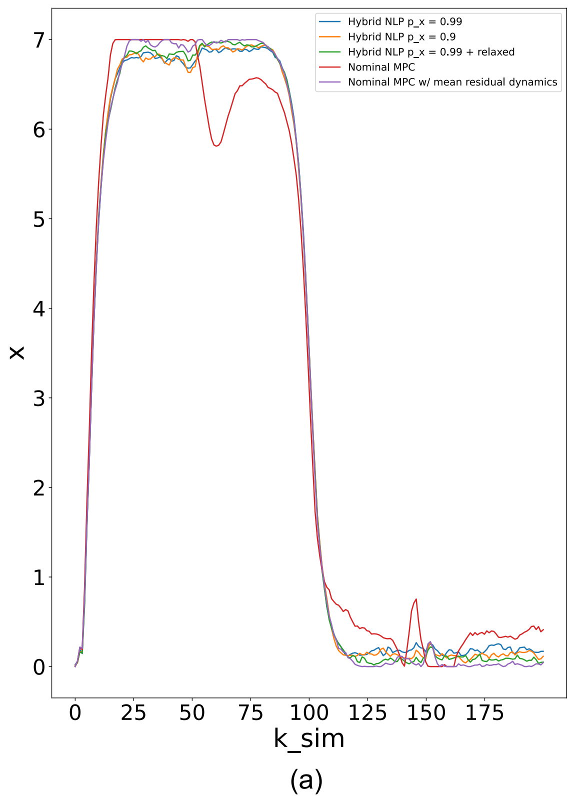
Unlike with the example in Section 3.7.1 where it is was desired to measure the extent of constraint violation, in this section I will instead clip any illegal state transitions (i.e., transitions to states $y_k^\delta \notin \cup_{i=1}^{\mathcal{R}} R_i = \Delta_{in}$) sampled from the stochastic environment to within the specified bounds. Hence, only the number of constraint violations are counted. This is in keeping with practical situations where the original constraints might be indicative of walls or track boundaries where any violation is bad, regardless of the extent.

Note. Practically, there might be collision / contact / rebound dynamics that must be modelled in the event of illegal state transitions but for the sake of simplicity these are not considered here.

The proposed methodology is tested on a boundary tracking case to demonstrate approximate constraint satisfaction while maximizing closed-loop performance.

Tracking waypoints on the boundary of the workspace

In this task, a set of high-level waypoints $W = [(7,0),(7,7),(0,7),(0,0)]$ is considered denoting the vertices of the 2-D quadrotor's workspace. The closed-loop simulation is run for $N_{sim} = 200$ timesteps determined in advance. In accordance with Remark 3.6.2 nominal MPC is used to generate a reference trajectory, x^{ref}, u^{ref}, with the cost defined by shifting the waypoint in W to be tracked at every $N_{sim}/N_W = 200/4 = 50$ timesteps. x^{ref}, u^{ref} are then utilized to compute the quantities in Algorithm 1 and warmstart [139] the online controller.



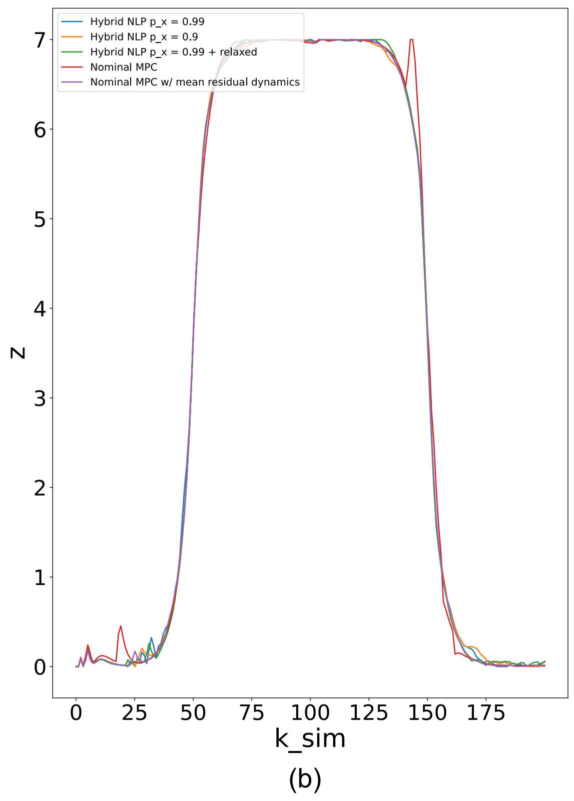


Figure 3.13: A plot depicting an example of the closed-loop $x - z$ trajectories obtained for each controller i.e., nominal MPC, nominal MPC with mean residual dynamics, hybrid GP-MPC with p_x=0.99, 0.9 and hybrid GP-MPC with $p_x = 0.9$ and "relaxed" shrinking constraints.

The approach is demonstrated for $p_x = \{0.99, 0.9\}$ (p_u is neglected since $K = 0$ as described in Section 3.6.2). Also, since Boole's inequality can yield a conservative approximation of the joint chance-constraint at every timestep, I also consider the use of a slightly "relaxed" version of the inverse Gaussian CDF constraint involved in (3.16a). This utilizes the fact that the residual dynamics in this example are limited to affect x. I make the assumption that the dynamic coupling between the state variables is small enough for uncertainty in state variables other than x to be neglected. Hence, I make the choice of $\bar{p} = 1 - \frac{1-p_x}{2}$ for the individual chance constraints derived from the joint. This test is run for $p_x = 0.9$.

A quantitative comparison of the closed-loop cost (3.28) and constraint violation (3.29) is shown in Table 3.5. A qualitative comparison of the closed loop trajectories in the $x - z$ plane is shown in Figure 3.13.

Analysis of results. As expected, the controller with only nominal dynamics has bad closed-loop performance and significantly violates the constraints (regardless of whether $p_x = 0.99$ or $p_x = 0.9$) and has large variance between runs. The results for the controller that incorporates only mean dynamics is indicative of how one might expect a controller using a deterministic (but still hybrid) residual model to perform. It seemingly demonstrates the best closed-loop cost but this is due to the *lack of rebound dynamics*. It also exceeds the constraint violation limits. It can be seen from Figure 3.13 that all of the proposed NLP-based GP-MPC controllers perform reasonably well and Table 3.5 does confirm the hypothesis that using a “relaxed” version of Boole’s inequality boosts performance. Moreover, it actually performs better in terms of constraint violation in this particular scenario. This could be due to the fact that it allows closer tracking of the provided reference trajectory allowing for the shrunk sets to remain more accurate over the closed-loop simulations.

Method	$\hat{\delta}_x$ (mean)	μ^c_{cl}	Σ^c_{cl}
Nominal MPC	0.178	272.80	1.9×10^3
Nominal MPC w/ $\mu^{\hat{g}}_k$	0.111	**34.46**	0.655
Proposed GP-MPC ($p_x = 0.99$)	0.010	66.17	6.93
Proposed GP-MPC ($p_x = 0.9$)	0.012	54.15	**6.52**
Proposed GP-MPC ($p_x = 0.99$ + “relaxed”)	**0.006**	**48.22**	7.54

Table 3.5: Empirically measured average constraint violation for the closed-loop ($\hat{\delta}_x$) trajectory along with mean and variance of the closed-loop cost. Results in red violate the chance constraints and those highlighted in bold indicate they are the best in comparison to other approaches. Computed over 50 simulation runs.

Run-time comparison

Since the code is implemented in Python, the run-times are slower than what might be expected of a typical C/C++ implementation generated using a proprietary software like e.g., FORCESPro [147]. As a result, the run-time of the online controller (3.26) is compared against a nominal online MPC controller. Simulations were run on a i7-12700H CPU. As shown in Table 3.6, the runtimes are significantly lower compared to the original MINLP controller and relatively close to the nominal MPC runtimes.

Remarks.

Method	μ_{run} (ms)	σ_{run} (ms)
Nominal MPC	137.8	4.16
Proposed NLP GP-MPC	176.09	7.9
MINLP GP-MPC	Timeout	Timeout

Table 3.6: Run-time statistics comparison between a nominal MPC controller and the proposed parametrized NLP GP-MPC controller computed over 25 runs. μ_{run}, σ_{run} denotes the run-time mean and variance respectively.

- Comparing (3.26) with a typical nominal MPC controller, it is clear to see that the increase in run-time is due to the mean dynamics that must be taken into account for the online GPMPC-based controller. Here, a relatively small dataset of ∼85 points is used for training for *each* mode of the hybrid GP model but in practice, sparse GP methods (Section 2.2.1) would need to be used to be able to expect the speed benefits demonstrated by the online GP-based NLP controller.

- The results of the controllers depicted here do not let the open-loop optimizations run to completion. This is because in practice, any added performance benefits are lost if the controller is unable to keep up with the desired sampling times. The open-loop optimizations are run for a maximum of 35 iterations but early termination is allowed if the trajectory obtained is below a constraint violation tolerance of 10^{-2} over the entire horizon N.

- For the simulations, the default linear solver of MUMPS [5] was used in the optimization. There could be potential speed-up benefits gained from using proprietary HSL solvers [120].

- In the code implementation, the residual dynamics are computed in parallel for all modes of the hybrid GP model. Due to this parallel prediction capability, it is not expected that run-times between the baseline and proposed GP-MPC approaches differ (except maybe for memory-constrained systems where the parallelization is not possible). In fact, the baseline could be worse off due to excessive constraint shrinking in complex tasks yielding more infeasible solves that run to a user-specified maximum number of iterations before terminating. However, this is not demonstrated here as the workspace considered here is relatively simple and neglects the inclusion of obstacles.

3.8 Conclusion

The proposed hybrid MPC controller developed using a dataset generated from a piecewise residual dynamics model is capable of solving Problem 2 while also demonstrating better performance over a baseline controller. The proposed method generalizes well to cases where the baseline GP-based modeling and control struggles to trade-off control performance and constraint satisfaction across a variety of scenarios. However, it exhibits significantly restrictive computation times of $\sim$51 seconds for open-loop optimization solves.

In contrast, the parametrized version of this controller reduces the problem to an NLP and demonstrates significant speed-up in computation time while still providing performance and safety benefits. However, the results indicate that the approach would benefit from the use of iterative risk allocation methods to reduce the degree of conservatism. Moreover, the parametrization would affect the optimality and robustness of the solution on more dynamic challenging tasks. It would beneficial to have multiple parametrized controllers running in parallel to track varying reference trajectories in order to recover some of the optimality of the MINLP formulation.

Chapter 4

Identifying region locations in unseen environments.

In Chapter 3, I made the assumption that the region locations in a particular environment are known. This is quite a limiting assumption when dealing with a system that needs to be transferred to a new environment where the region locations are apriori unknown. In this chapter, I deal with systems where the regions are controlled by the workspace variables and propose a methodology to learn a classifier capable of "mapping" points in the workspace to a corresponding terrain or mode of the piecewise residual model.

4.1 Motivating Example.

A modification of Example 3.1.1 can be used to motivate the relevance of the problem considered in this chapter.

Consider a wheeled robot operating in a workspace comprised of several different terrains (e.g., snow, asphalt). Assume that the robot is provided access with a probabilistic hybrid residual model comprised of modes corresponding to each of the terrains present in the workspace as depicted in Figure 4.1. Each of these residual models is an approximation of the unmodeled dynamics in play while operating in that specific terrain.

When initially placed in this workspace, the robot has no information about these regions. In other words, the unknown specifications are (i) the constraints that define the location of these regions and (ii) the mode of the hybrid residual model active in each of these different regions (which in this example corresponds to the terrain assigned to a

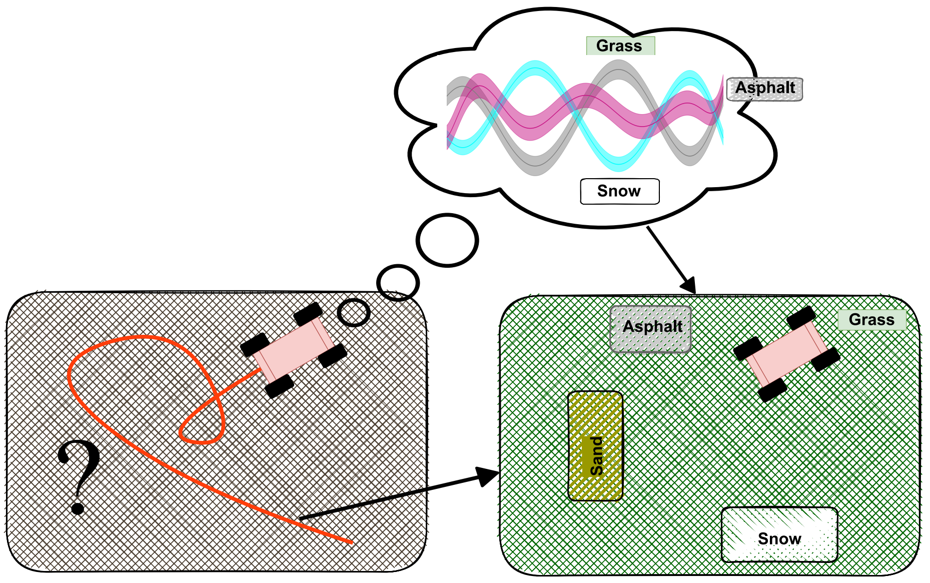


Figure 4.1: A wheeled robot with access to a hybrid residual model. Can it leverage known information along with measurements to build a "map" of how the terrains are distributed across the environment it is operating in?

particular region). This necessitates the use of a "mapping" algorithm to approximately infer region locations and their terrain assignments. This map information can then be used in a controller that uses terrain-specific dynamics to predict trajectories over a horizon into the future.

4.2 Problem Setup and Statement

Notation. In this chapter, I will retain the notation used in Chapter 3. As before $R_{\text{set}} = \{R_1, R_2, \dots R_{\mathcal{R}}\}$ partitions Δ_{in}. Here, however, I will make the assumption that the *workspace* variables control the region locations. I use $X_{ws} \subseteq X$ to denote the workspace for a given environment. Hence, $\Delta_{in} = X_{\text{ws}}$ and as shorthand the projection of the state onto the workspace is defined as, $\psi : X \to X_{\text{ws}}, \psi(x) = B_\delta x = x_{ws}$. ■

As before the workspace is partitioned into regions but here each region has a corresponding "terrain" assigned to it (similar to the brief discussion in Remark 3.5.6 but this is further formalized here). The word "terrain" will be used in this context to indicate a link to a specific mode of the piecewise residual dynamics and does not necessarily need to indicate surfaces like asphalt or snow as in Example 4.1. $R_{\text{ws}}(\cdot)$ can be defined as the function that maps $x_{ws} \in X_{\text{ws}}$ to the index of the region they belong to, i.e., $R_{\text{ws}} : X_{\text{ws}} \to R_{\text{set}}$. It is assumed the regions still adhere to Assumption 3.2.2 and hence $R_{\text{ws}}(\cdot)$ is defined $\forall x_{\text{ws}} \in X_{\text{ws}}$. Hence, I define a function $R : X \to R_{\text{set}}, R(x) = R_{\text{ws}} \circ \psi(x)$.

Assumption 4.2.1. *The terrains that might be encountered in the unknown environment are known in advance. Thus, $T_{set} = \{1, 2, \dots, \mathcal{T}\}$ can be defined where $\mathcal{T}$ is known in advance and use T_i to denote the terrain type corresponding to the i^{th} index in T_{set}.*

There exists a true underlying function $T_{\text{cat}}(\cdot) : R_{\text{set}} \to T_{\text{set}}$ that maps regions to terrain types for a given environment. $T_{\text{cat}}(\cdot)$ does not need to be surjective i.e., it is possible for a terrain not to be present in the environment. The function that maps states to terrains can then be defined as $T(\cdot) : X \to T_{\text{set}}$ as,

$$T(x) = T_{\text{cat}} \circ R(x) \tag{4.1}$$

With this construction, the subtle differences when compared to the problem setup in Chapter 3 are as follows,

- The system dynamics under consideration as similar to (3.1) with the discrete variables in (3.1c) being defined instead as,

$$\delta_{i,k} = \begin{cases} 1, & \text{if } T(x_k) = i \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

- The noise model in (3.2) now varies with the terrain index in T_{set}.

- The datasets collected (3.4) are now terrain-dependent as opposed to region-dependent allowing us to learn the various modes of the hybrid model as,

$$\hat{g}_{\text{set}} = \{\hat{g}_1, \hat{g}_2, \dots, \hat{g}_{\mathcal{T}}\} \tag{4.3}$$

Since $T(x_k)$ is a function of the environment, $\delta_{i,k}$ is as well. As a result, when operating in an unknown environment with workspace X_{ws}, it is required to learn an approximation of $T(x_k)$ as a pre-requisite to approximating the piecewise dynamics defined by (4.2). To do this, the following assumption is made.

Assumption 4.2.2. *The learnt models in $\hat{g}_{set}$ are capable of providing us with likelihoods with which a measured residual term, d_k, was generated by them. In other words, each of the models in $\hat{g}_{set}$ must be Bayesian and capable of generating $p(d_k \mid (x_k, u_k) \,\&\, T(x_k) = i)$.*

Since it is desirable that this approach work in general, i.e., regardless of the structure of the environment the system is placed in, the following condition on the approximated residual dynamics is enforced,

Remark 4.2.1. *The residual models in $\hat{g}_{set}$ are limited to depend on variables not included in X_{ws} i.e., $G_{in} \cap X_{ws} = \phi$. Figure 4.2 depicts the dependencies between the variables that determine the residual magnitude.*

Note. While the true residual dynamics in an environment might varying as a function of the workspace variables, training the models in $\hat{g}_{\text{set}}$ to depend on them implies that they will over-fit to the specific environment where the samples are collected, thus harming generalization to unknown environments. As a result, Remark 4.2.1 holds practical merit and is not particularly limiting *unless* the residual models in g_{set} are desired to be fine-tuned online to the new environment in a streaming-based [40, 31, 109] fashion (a relevant problem that is not considered in this thesis).

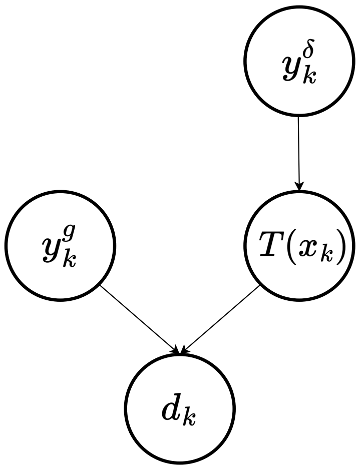


Figure 4.2: A directed graph showing dependence relations between the different variables affecting the residual magnitude.

By Bayes' Rule,

$$p(d_k \mid y_k^\delta, y_k^g, T(x_k) = i) = \frac{P(T(x_k) = i \mid y_k^\delta, y_k^g, d_k)P(d_k \mid y_k^\delta, y_k^g)}{P(T(x_k) = i \mid y_k^\delta, y_k^g)} \tag{4.4}$$

$$P(d_k \mid x_k, u_k) = P(d_k \mid y_k^\delta, y_k^g) \tag{4.5a}$$

$$= \sum_{i=0}^{\mathcal{T}} P(d_k \mid y_k^g, T(x_k) = i) P(T(x_k) = i) \tag{4.5b}$$

which is constant $\forall i \in T_{\text{set}}$.

This yields the posterior update rule,

$$P(T(x_k) = i \mid (x_k, u_k), d_k) \propto p(d_k \mid y_k^\delta, y_k^g, T(x_k) = i) P(T(x_k) = i \mid y_k^\delta, y_k^g) \tag{4.6}$$

where $p(d_k \mid y_k^\delta, y_k^g, T(x_k) = i)$ indicates the likelihood term and $P(T(x_k) = i \mid y_k^\delta, y_k^g)$ indicates the prior.

Given that Assumption 4.2.2 holds, and given a dataset $D_{\text{unk}} = \{((x_k, u_k), d_k)\}$, obtained online in an unknown environment (using the procedure outlined in (3.3)), each sample can be associated with a vector,

$$p(\hat{T} \mid x_k, u_k, d_k) = \begin{bmatrix} p(T(x_k) = 1 \mid ((x_k, u_k), d_k) \\ p(T(x_k) = 2 \mid ((x_k, u_k), d_k) \\ \vdots \\ p(T(x_k) = \mathcal{T} \mid ((x_k, u_k), d_k) \end{bmatrix} \tag{4.7}$$

An augmented dataset can now be defined as,

$$D_{\hat{T}} = \{(x_k \, , \, p(\hat{T} \mid x_k, u_k, d_k)) \mid ((x_k, u_k), d_k) \in D_{\text{unk}}\}$$

can then be defined. Without loss of generality, it can be assumed that

$$\sum_{i=1}^{\mathcal{T}} p(T(x_k) = i \mid (x_k, u_k), d_k) = 1 \tag{4.8}$$

which can be achieved by any normalization technique.

This dataset can be used to train an approximation, $\hat{T}(x)$, to (4.1). This in turn approximates the discrete variable mapping (4.2) as,

$$\hat{\delta}_{i,k} = \begin{cases} 1, & \text{if } (\hat{T}(x_k)) = i \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

Remark 4.2.2. *In general, if $\hat{T}(x_k)$ yields a vector that is not a one-hot encoding but rather logits or a probability distribution, then an argmax operator needs to be applied to the vector to generate the one-hot encoding.*

Problem 3. *Given data points, D_{unk} collected in an environment with an unknown distribution of terrains across its workspace, construct an augmented dataset, $D_{\hat{T}}$, containing labels as in (4.7). Train a classifier using $D_{\hat{T}}$ to obtain a learnt approximation, $\hat{T} : X \to T_{set}$, of the true function $T(x)$ in (4.1).*

Note. Problem 3 is formulated to directly learn $\hat{T}$ over the domain, X. This is in contrast to learning an approximation of each of the two functions that constitute the composition in (4.7) which necessitates the more restrictive assumption that the number of regions, $\mathcal{R}$, in the environment is known (and also requires an additional δ array over regions as described in Remark 3.5.6 and the discussion that follows it).

The true piece-wise residual dynamics for an unknown environment can now be approximated as,

$$x_{k+1} = f(x_k, u_k) + \hat{g}(x_k, u_k) \tag{4.10a}$$

$$\text{where, } \hat{g}(x_k, u_k) = \sum_{i=1}^{\mathcal{T}} \hat{\delta}_{i,k} \hat{g}_i(x_k, u_k) \tag{4.10b}$$

With the approximate learnt dynamics in (4.10), it is desired to solve the following optimal control problem below.

Problem 4. *Given a task defined by a set of high-level waypoints develop an iterative learning-based planner-controller architecture for the system (3.1) that iteratively collects trajectory data while trying to track a reference trajectory x^{ref} and solves Problem 3 to improve the estimate of $\hat{T}$. The architecture aims to reduce the expected cost $J = \sum_k \mathbb{E}_{w_k}(||x_k - x_k^{ref}||_Q^2 + ||u_k||_R^2)$ over time while adhering to the chance-constraints in the limit when $\hat{T}(x) \to T(x) \; \forall \; x \in X$.*

4.3 Methodology

4.3.1 Training a classifier on a dataset of soft label vectors.

When working with GPs, (2.7) tells us that the output for a given set of inputs is a multivariate Gaussian distribution. For a given terrain-specific mode of the hybrid model, rewriting (2.7) using a negative log likelihood yields,

$$\mathcal{L}(\mu_k^{\hat{g}_i}, \Sigma_k^{\hat{g}_i}) = \frac{1}{2}\left[(x_k - \mu_k^{\hat{g}_i})^T {\Sigma_k^{\hat{g}_i}}^{-1}(x_k - \mu_k^{\hat{g}_i}) + n_d \log(2\pi) + \log\left(\det(\Sigma_k^{\hat{g}_i})\right)\right] \tag{4.11}$$

Given this, it is relatively straightforward to estimate the likelihood that a measured residual was generated by each of the modes in the learnt hybrid GP model as depicted in Figure 4.3.

Assumption 4.3.1. *For simplicity, the prior term* $(p(T(x_k) = i \mid (x_k, u_k)))$ *in* (4.6) *is a static uninformative (uniform) prior across terrains that does not change across successive iterations of the algorithm.*

This yields,

$$P(T(x_k) = i \mid (x_k, u_k), d_k) \propto p(d_k \mid y_k^{\delta}, y_k^{g}) \tag{4.12}$$

which allows (4.7) to be obtained.

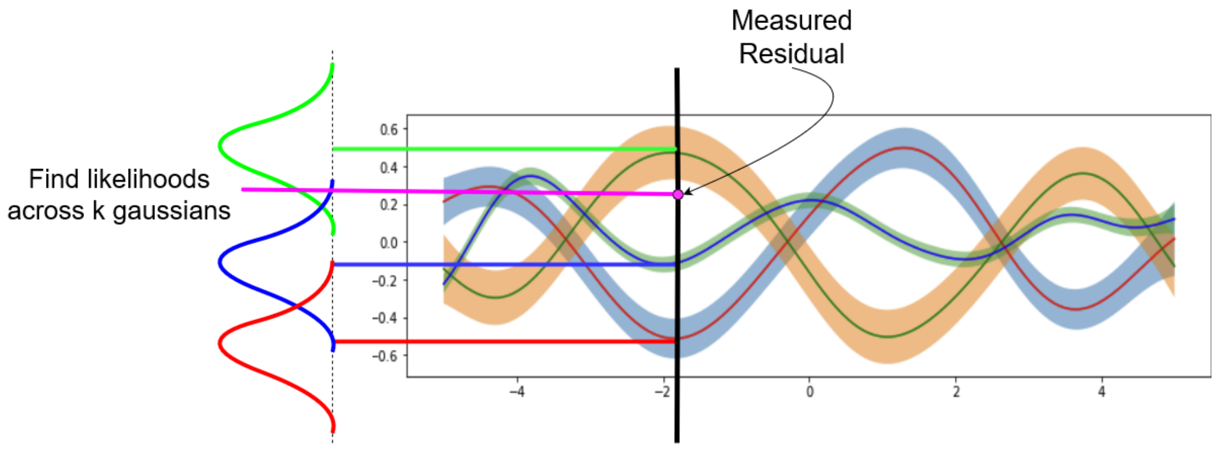


Figure 4.3: A visual depiction of the process of computing likelihoods that a particular measured residual was generated by one of the learnt modes of the hybrid model for a given input.

As the title suggests, it would be desirable to have a classifier capable of training on (and hence predicting) soft labels for 2 main reasons viz.,

- The hybrid GP model naturally outputs soft label likelihood vectors. Training on these directly as opposed to converting them to a one-hot vector encoding prevents learning over-confident assignments as demonstrated in Example 4.3.2.
- The ability to treat soft labels as viable priors to substitute in (4.6). This use case is not considered in this thesis.

For simplicity in this thesis, I choose to use a simple single-layered NN as the classifier.

I now present an algorithm to perform terrain mapping across the workspace of an unknown environment by training a classifier every time new information, encapsulated in the form of a trajectory executed in closed-loop, is received.

Algorithm 2 Iterative terrain mapping

```
Require: $f$: Nominal dynamics
Require: $\hat{g}_{\text{set}}$: Hybrid residual model
Require: $D_{in}$: Accumulated dataset
Require: $x, u$: New trajectory to be manipulated and added to the dataset
Require: $B_{g,in}$: Matrix defining projection onto GP input space
Require: $\psi$: Function to project $x$ onto $X_{ws}$
 1: procedure SimulatedAnnealing
 2:   $N_{sim} \leftarrow len(x)$                                  ▷ Trajectory length
 3:   $y^{cl} \leftarrow \emptyset$                                ▷ Classifier training labels
 4:   $y^{\delta} \leftarrow \emptyset$                            ▷ Classifier training inputs
 5:   foreach $k \in \text{range}(N_{sim})$ do
 6:     $d_k \leftarrow$ ComputeResidual$(f, x_k, u_k, x_{k+1})$   ▷ As in (3.3)
 7:     $z_k \leftarrow [{x_k}^T, {u_k}^T]^T$
 8:     $y^{\delta} \leftarrow y^{\delta} \cup \psi(x_k)$         ▷ Append new training input.
 9:     $y^g_k \leftarrow B_{g,in} z_k$
      // Append new training label.
10:     $y^{cl} \leftarrow y^{cl} \cup$ Softmax(ComputeLikelihood$(y^g_k, d_k, \hat{g}_{\text{set}})$)
    // Concatenate new dataset with existing one accumulated over previous runs.
11:   $D_{\text{new}}[x] \leftarrow y^{\delta}$
12:   $D_{\text{new}}[y] \leftarrow y^{cl}$
13:   $D_{\hat{T}}[x] \leftarrow D_{\text{in}}[x] \cup D_{\text{new}}[x]$
14:   $D_{\hat{T}}[y] \leftarrow D_{\text{in}}[y] \cup D_{\text{new}}[y]$
15:   model $\leftarrow$ TrainMappingClassifier$(D_{\hat{T}})$
16:   return model
```

The residuals on line 6 are computed as described in (3.3). The projection function on line 8 is of note as it reduces the dimension of input that the classifier is to be trained on by pruning redundant (for the purpose of terrain "mapping") variables in the state vector and limiting the consideration to $y^{\delta}_k = \psi(x_k) = x_{ws} \in X_{ws} = \Delta_{in}$. The likelihood function on line 10 requires computing (4.11) across all terrains and then stacking them together in a vector. Passing this vector through the softmax allows the requirement in (4.8) to be met. Finally, I now describe the loss function at the core of training the NN classifier mentioned on line 16.

Loss function for the NN classifier.

Training the classifier requires a loss function capable of accounting for soft label probability distributions generated by (4.8). Cross-entropy loss naturally lends itself to this while also being able to handle for one-hot labels (as is what it is typically used for). The classification problem considered has $\mathcal{T}$ possible class assignments corresponding to each of the terrains that might be encountered in the environment. For simplicity, the binary (2-class) cross entropy loss is considered which can straightforwardly be extended to the multi-class case.

For discrete distributions p and q, with $\mathcal{T}$ possible assignments, cross entropy is defined as,

$$H(p, q) = -\sum_{i=0}^{\mathcal{T}} p(i \mid x_k) \log q(i \mid x_k) \tag{4.13}$$

When used with one-hot vectors, p is a distribution where the probability is 1 for the observed terrain and 0 otherwise while q is representative of the discrete probability distribution learnt by the classifier across the domain of the input space. p reduces to an indicator function for the corresponding class as a result yielding,

$$H(p, q) = -\log q(i \mid x_k) \tag{4.14}$$

where $i \in T_{\text{set}}$ is such that $p(i \mid x_k) = 1$.

When dealing with soft label assignments to p, such a simplification cannot be made. This is the case I will be working with (further motivated in Section 4.3.2) as defined by (4.7), (4.11). Given that the labels are normalized (4.8), (4.13) can be used to yield,

$$H(p, q) = -(p(i = 0 \mid x) \log q(i = 0 \mid x) + p(i = 1 \mid x) \log q(i = 1 \mid x)) \tag{4.15}$$

Note. A third alternative would be an intermediate between these approaches that uses one-hot vector encoding but uses a dynamic weighting for the loss on each sample based on a distance metric (e.g., (4.19)) that measures the similarity between distributions generated by each mode. This is not considered in this thesis.

A typical training loop with cross-entropy loss and gradient back-propagation using optimizers like [78] can now be used to train an NN classifier.

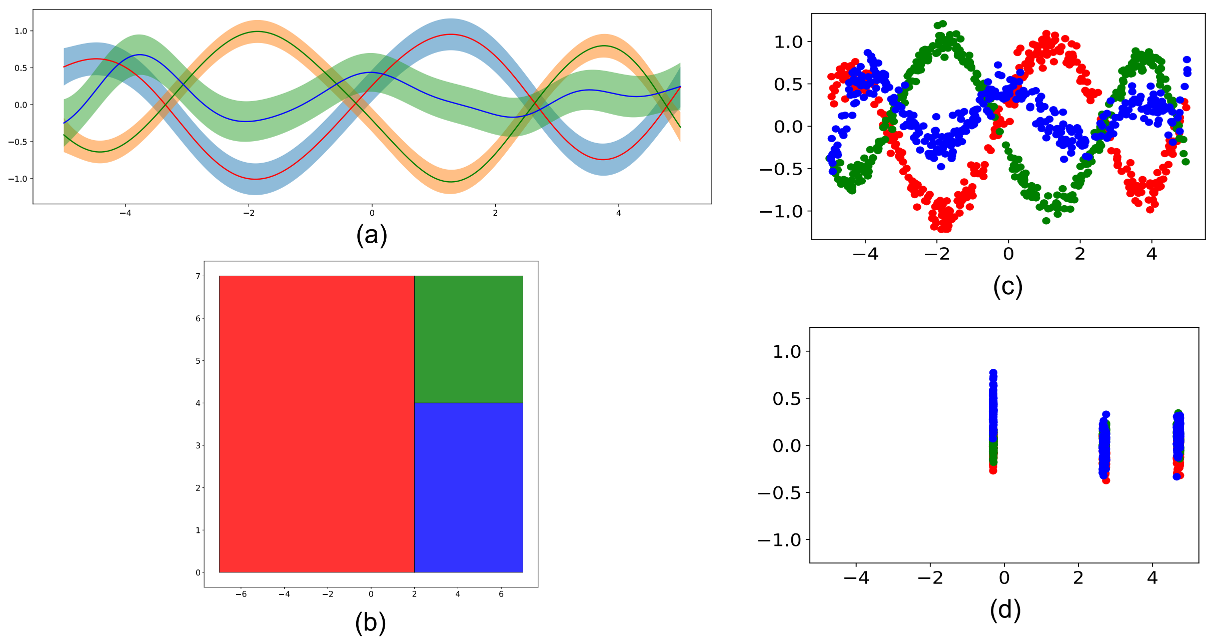


Figure 4.4: (a) Trained hybrid GP model. (b) Ground truth regions spread across workspace. (c)/(d) Example of uniform random sampling and biased sampling across G_{in} respectively.

4.3.2 Soft vs. Hard label training datasets

I consider an example that demonstrates the benefits of training on soft labels. First, assignments to y_k^δ are sampled uniform randomly across the workspace. I then consider two cases for sampling assignments to y_k^g. The first case considers uniform random sampling across the GP input space and the second considers sampling biased towards the regions of the GP input space where there is a high degree of overlap between the Gaussian distributions output from 2 or more modes of the hybrid model. The problem setup, along with examples of the 2 sampling schemes, is depicted in Figure 4.4. A simple single-layered NN classifier with 32 hidden nodes in the hidden layer is used for these experiments.

While the cross-entropy loss is most natural to measure test accuracy, the metric I use to measure performance involves converting the predictions to hard labels and counting the number of samples correctly classified. This is motivated by the fact that the controller architecture currently cannot handle for probabilistic distributions over regions while meet-

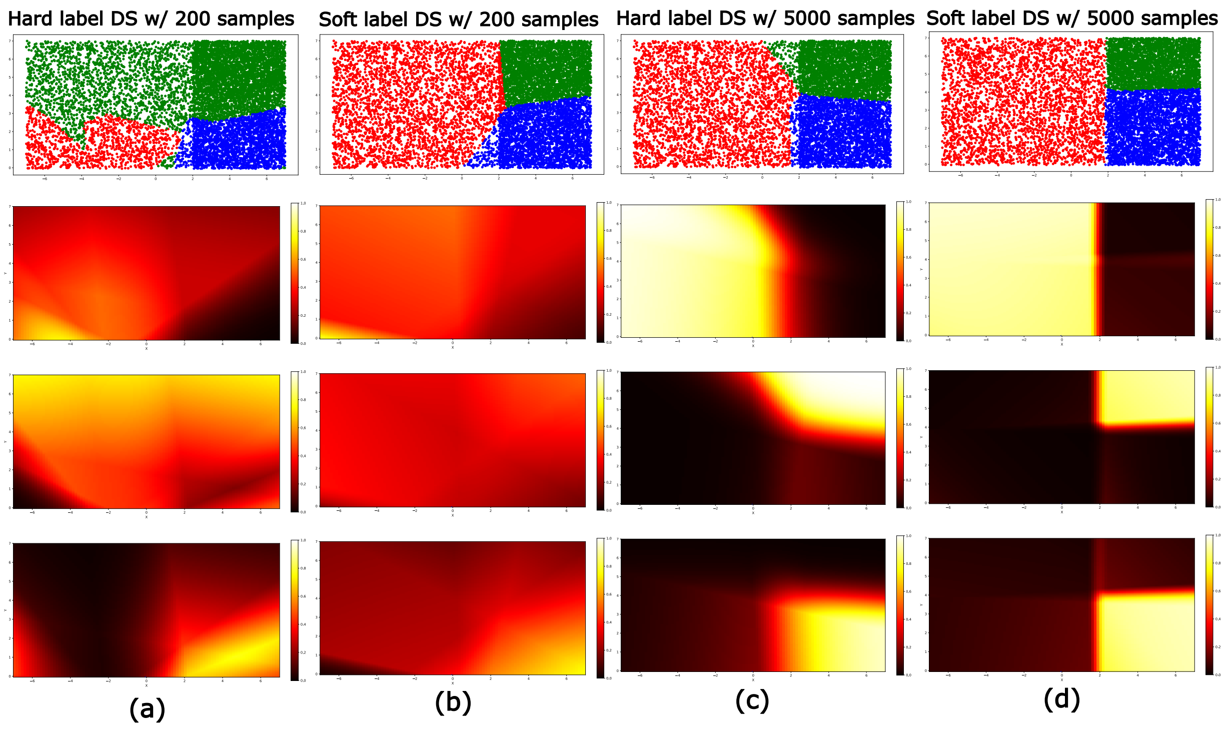


Figure 4.5: Plots depicting the qualitative outputs on a test dataset after training classifiers using the soft (b)/(d) and hard label (a)/(c) approaches when working with a random sampled dataset. (a)/(b) uses 200 training samples and (c)/(d) uses 5000 training samples. Row 1 corresponds to the argmax predictions. Row 2, 3 and 4 correspond to classifier logit outputs for each region after applying the softmax operator.

ing the safety constraints. As such, the classifier output must be converted to hard labels for use in a $\delta-$parametrized NLP controller. Hence, the metric that counts the number of correctly classified samples is more relevant.

For test examples, Figure 4.5 visualizes the performances of the classifiers on a dataset constructed by random sampling across G_{in}. Figure 4.6 shows heatmap outputs obtained after training the 2 classifiers on a biased dataset of 500 points are shown. The bias here is towards points in G_{in} where there is high overlap between the distributions output from each mode of the hybrid GP (as indicated in Figure 4.4 (d)). Finally, Figure 4.7 shows accuracy plots of both soft and hard label classifiers for each of the sampling schemes.

Remarks.

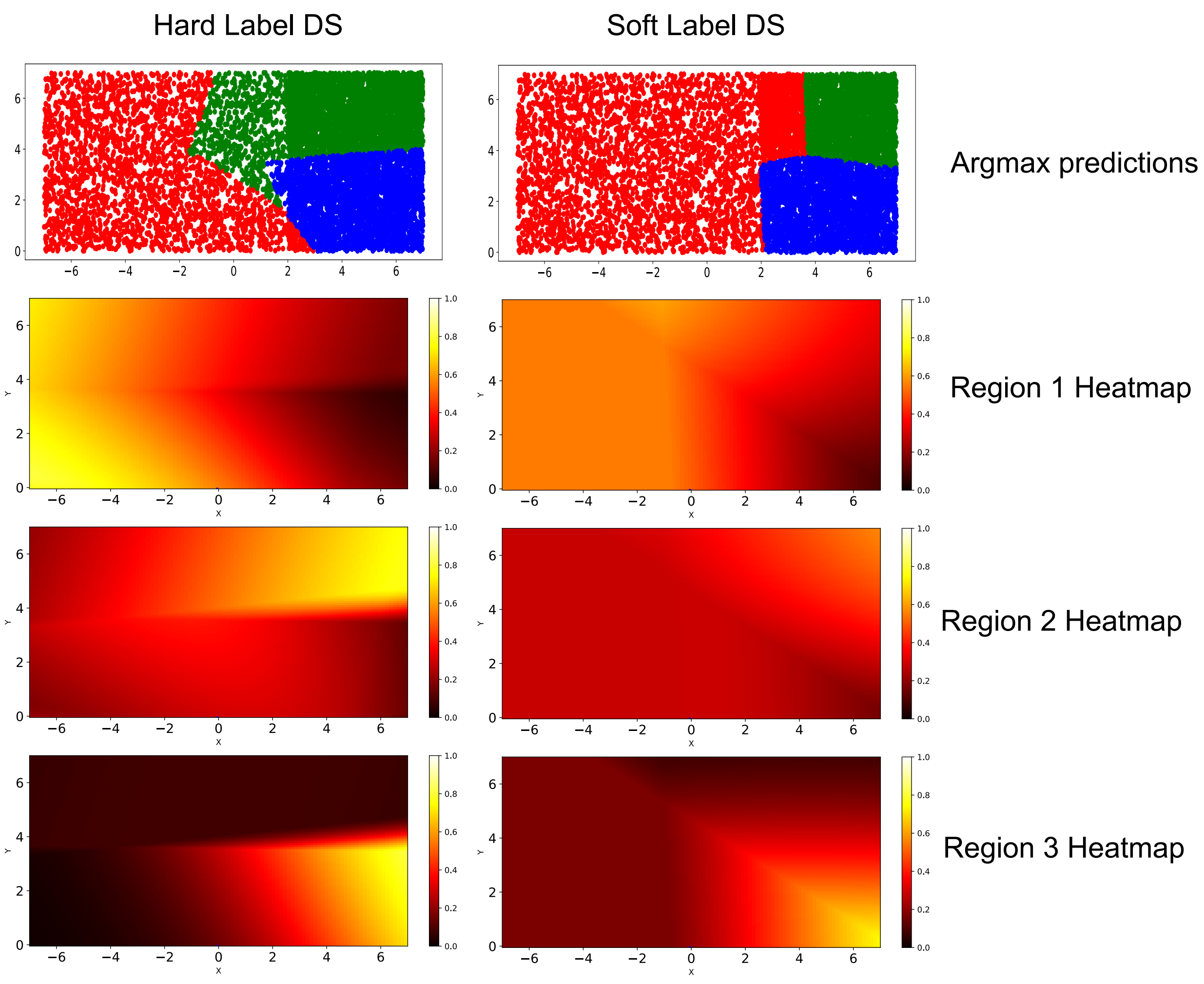


Figure 4.6: Qualitative outputs on a test dataset after training classifiers using the soft (b) and hard label (c) approaches when working with a biased dataset.

- For the case of random sampling across G_{in}, Figure 4.5 shows that the soft label approach is rightly conservative when dealing with fewer data points. It also learns much cleaner boundaries when compared with the hard label approach as the number of data points increases.
- For the case of biased sampling, depicted in Figure 4.6, it is clear that the soft label method still proceeds with a degree of conservatism whereas the hard label approach

does assign higher probability magnitudes and fails to perform as well due to nearby samples with conflicting labels. This is confirmed by the accuracy plot in Figure 4.7.

- From Figure 4.7, it is clear that both approaches work well with random sampling particularly as the dataset size increases. However, the soft label approach works much better than the hard label approach when looking at the mean training accuracy particularly in the case of the biased training set and working with fewer samples.

- Though the error bars are large across both, this is partially due to bad weight initialization, for a relatively simple NN classifier, leading to convergence to a poor local optimal model. The mean accuracy being closed to the upper limit indicates, however, that re-initializing weights usually fixes this issue. Training multiple randomly initialized models in parallel and selecting one with the lowest loss would work in practice and this will be utilized .

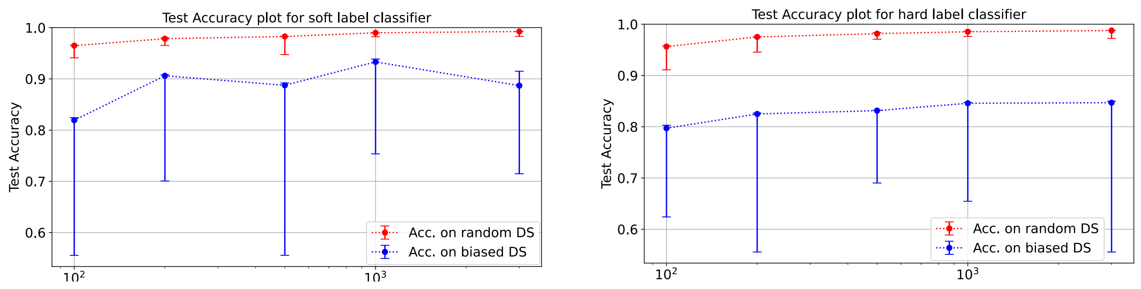


Figure 4.7: A graph showing accuracy with error-bars of both (a) soft and (b) hard label trained classifiers as a function of the number of training samples ([100, 200, 500, 1000, 3000]).

4.3.3 Terrain mapping planner-controller architecture

The resulting architecture is quite similar to the approach proposed in Section 3.6.2. The only differences are as follows,

- In Line 15 of Algorithm 1, the condition $H_j y^{\delta}_{k_{sim}} - b_j \leq \mathbf{0}$, is instead replaced with $\hat{T}(x_{k_{sim}}) = j$ since the polytopes defining the regions are no longer assumed to be known. As a result, the discrete variables returned on Line 25 are $\hat{\delta}^{ref}$ instead of δ^{ref} to reflect the approximation.

- In the parametrized NLP controller, the residual mean dynamics (3.26c) are instead replaced by,

$$\mu_k^{\hat{g}} = \sum_r \hat{\delta}_{r,k}^{ref} \mu_k^{\hat{g}_r}$$

to reflect the incorporation of the approximate discrete variable assignments into the dynamics prediction.

Remark 3.6.1 is now updated as follows,

Remark 4.3.1. *Let $i \in T_{set}$ be the index of the terrain* assumed *to be active at timestep k, i.e., $\hat{T}(x_{k+k_{sim}}^{ref}) = i$. The controller in (3.24) does* not *enforce constraints of the form $\hat{T}(\mu_k^x) = i$.*

In addition to the discussion that follows Remark 3.6.1 there is another limitation with using this approximate mapping classifier $\hat{T}(\cdot)$ that prevents us from enforcing this constraint. Unlike a constraint of the form $Hx \leq b$ where it is straightforward to quantify the degree of constraint violation and compute the constraint jacobian (purpose as described in Section 2.2.1), in light of Remark 4.2.2, a constraint $\hat{T}(\mu_k^x) = i$ can no longer work with gradient-based optimizers due to the sharp switching transitions induced by changes between one-hot encoding vectors.

4.4 Data-efficient mapping of unknown environments.

Given a state, $x_k \in X$, the set of distributions output by each mode of the hybrid model $\hat{g}(x_k)$ can be formalized as,

$$\mathcal{N}_{\text{set}}(y_k^g) = \{\mathcal{N}(\mu_k^{\hat{g}_t}, \Sigma_k^{\hat{g}_t}) \ \forall\, t \in T_{\text{set}}\} \tag{4.16}$$

What kind of efficiency is being considering? Depending on the trajectory to track, the samples collected might be in parts of G_{in} where there is a high degree of overlap between two or more distributions in $\mathcal{N}_{\text{set}}$. In other words, for a given d_k collected in these parts, the likelihood vectors constructed using (4.11) can have 2 or more entries that are close to each other. This can significantly affect the convergence of the classifier as motivated by the accuracy plot for the biased sampling case in Figure 4.7.

To address this problem, it could be desired to initially collect some samples in a manner that allows us to better understand how terrains are distributed across the path to be tracked. This can be considered as an "exploratory" phase limited to a small area of the workspace around the trajectory that is desired to be tracked. As a result, the "efficiency" considered here is linked to building a more accurate dataset when incorporating such "exploration" when compared to approaches that do not utilize the proposed methodology.

To achieve this, I formulate an optimization problem capable of identifying points across G_{in} that maximize the "distance" between the distributions in $\mathcal{N}_{\text{set}}$. Given the output of this optimization, x^{opt}, a term of the form shown below can then be added to the stage cost function of the optimization when using a *nominal* MPC to collect samples for $D_{\hat{T}}$ when there is an initial lack of data points.

$$C_{expl} = \sum_{k=0}^{N} \min\{w_p \|x_k - x_p^{opt}\|^2_{Q_{expl}} \; \forall \; p \; \in \{1, \ldots, \text{len}(x_{opt})\} \tag{4.17}$$

Many auto-differentiation engines (including Casadi [7] which is used for the experiments) implement a differentiable version of the *min* operator in (4.17) for use in these optimizations.

Remark 4.4.1. *The minimization operator helps prevent heavily biasing the optimization towards certain parts of the state space when including (4.17) in the cost function. When assigning $w_p = c \; \forall \; p \; \in \{1, \ldots, len(x^{opt})\}$, this boils down to saying that the optimization should be indifferent about which point in x^{opt} to track.*

In Algorithms 3 and 4 the building blocks of the optimization problem referenced above are described and then brought together in the final solution implemented in Algorithm 5.

Building blocks of the optimization problem. A version of simulated annealing is presented in Algorithm 3. As a brief summary,

- Simulated annealing is a stochastic optimization method used to find near-optimal solutions to complex optimization problems where conventional gradient-based approaches might fail to perform. It starts with an initial guess x_{init} and iteratively explores the solution space by generating a new candidate solution (x_{new}) from the guess considered in the previous iteration (x_{current}).
- x_{new} is generated by a neighbourhood function, NeighbourhoodFn, as shown on Line 9. This biases the location of x_{new} based on some probability distribution (usually Gaussian) centred at x_{current}.
- The "goodness" of (ordering over) solutions is defined based on a heuristic energy function as introduced on Line 10. A lower energy function indicates a better solution.

- Better solutions are accepted immediately as on Line 13. Worse solutions are also accepted probabilistically based on the value of a temperature parameter, T, and the extent to which the new solution is worse than the current (ΔE). This is as implemented in Lines 19-22.

- The algorithm uses a cooling schedule, as shown on Line 26, to gradually reduce the exploration intensity. This allows it to escape local sub-optimal solutions initially, while allowing convergence towards the end of $k \in range(N_{iter})$

Clearly the functions ComputeEnergy and NeighbourhoodFn are at the heart of this algorithm and so their implementations for the optimization problem are now discussed.

ComputeEnergy - Distance metrics over Gaussian probability distributions. Defining an objective (energy) function for this optimization requires a metric capable of measuring the similarity between the distributions in $\mathcal{N}_{\text{set}}$. It would not make sense for the objective function to depend on the ordering in T_{set}. As a result, I opt to use the *symmetric* Bhattacharya distance metric.

The formula for Bhattacharya distance between two probability distributions p and q is given by:

$$D_{\text{B, gen}}(p, q) = -\int_{\mathcal{X}} \sqrt{p(x) \cdot q(x)}\, dx \tag{4.18}$$

Letting $\mathcal{N}_i$ denote the i^{th} element of $\mathcal{N}_{\text{set}}$ and given (2.7), (4.18) simplifies to the pairwise distance,

$$D_{\text{B}}(\mathcal{N}_i, \mathcal{N}_j) = \frac{1}{8}(\mu_i - \mu_j)^T(\Sigma_i + \Sigma_j)^{-1}(\mu_i - \mu_j) + \frac{1}{2}\ln\left(\frac{\det(\Sigma_{ij})}{\sqrt{\det(\Sigma_i) \cdot \det(\Sigma_j)}}\right) \tag{4.19}$$

where $\Sigma_{ij} = \det\left(\frac{\Sigma_i + \Sigma_j}{2}\right)$. (4.19) requires that $\mathcal{N}_i, \mathcal{N}_j$ have non-singular, p.s.d. covariance matrices which holds by Remark 2.1.2 and the fact that process noise will ensure that Σ_d is non-zero for all residual dimensions for any input. Smaller values of D_{B} are representive of a smaller distance and hence a higher degree of similarity between $\mathcal{N}_i, \mathcal{N}_j$.

The objective function for the optimization that aims to find points of maximum similarity can now be defined as,

$$\textsc{ComputeEnergy}(z_k) = \sum_{i=1}^{T} \sum_{j=i+1}^{T} D_{\text{B}}(\mathcal{N}_i(z_k), \mathcal{N}_j(z_k)) \tag{4.20}$$

Algorithm 3 Simulated Annealing

```
Require: $T$: Initial temperature
Require: $c\_f$: Cooling factor
Require: $\sigma$: Step size
Require: nbhd_args: Additional arguments for the neighborhood function
Require: energy_args: Additional arguments for the energy function
Require: $x_{\text{init}}$: Initialization point
Require: $N_{\text{iter}}$: Number of iterations to run for.
 1: procedure SimulatedAnnealing
 2:   $x_{\text{current}} \leftarrow x_{\text{init}}$
 3:   $x_{\text{best}} \leftarrow x_{\text{current}}$
 4:   $T_{\text{current}} \leftarrow T$
 5:   $E_{\text{current}} \leftarrow$ ComputeEnergy($x_{\text{current}}$, energy_args)
 6:   $E_{\text{best}} \leftarrow E_{\text{current}}$
 7:   foreach $k \in$ range($N_{\text{iter}}$) do
 8:     update $\leftarrow 0$
 9:     $x_{\text{new}} \leftarrow$ NeighbourhoodFn($x_{\text{current}}, \sigma$, nbhd_args)
    // Heuristic quantifying solution “goodness”
10:     $E_{\text{new}} \leftarrow$ ComputeEnergy($x_{\text{new}}$, energy_args)
11:     $\Delta E \leftarrow E_{\text{new}} - E_{\text{current}}$
12:     if $\Delta E < 0$ then
13:       $x_{\text{current}} \leftarrow x_{\text{new}}$
14:       update $\leftarrow 1$
15:       if $E_{\text{new}} < E_{\text{best}}$ then
16:         $x_{\text{best}} \leftarrow x_{\text{new}}$
17:         $E_{\text{best}} \leftarrow E_{\text{new}}$
18:     else
19:       $p \leftarrow \exp(-\Delta E / T_{\text{current}})$
20:       $r \leftarrow$ UniformRandomSample$(0, 1)$
21:       if $r < p$ then
22:         $x_{\text{current}} \leftarrow x_{\text{new}}$
23:         update $\leftarrow 1$
24:     if update $= 1$ then
25:       $E_{\text{current}} \leftarrow E_{\text{new}}$
26:     $T_{\text{current}} \leftarrow T_{\text{current}} \times c\_f$
27:   return $x_{\text{best}}$
```

where the summation bounds are obtained using the fact that $D_{\mathrm{B}}(\mathcal{N}_i, \mathcal{N}_i) = 0$ and the property that D_{B} is symmetric. In case it is desired to find points of *minimum* similarity, (4.20) simply needs to be negated.

NeighbourhoodFn - Sampling new candidate solutions in the GP input space. The neighbourhood function for a vanilla implementation of simulated annealing only takes in parameters $x_{\text{current}}, \sigma$ to sample x_{new} from the Gaussian distribution $\mathcal{N}(x_{\text{current}}, \sigma^2)$ as on Lines 5-6. However, I propose a modified approach in Algorithm 4 which uses 3 additional parameters

- An array of solutions, provided as input, accumulated from previous iterations of simulated annealing, solution_arr.
- A redundancy radius parameter used to prevent x_{new} from being within an ϵ-ball around any previously found solution in solution_arr as done on Lines 7-15.
- A bounded GP input space, G_{in} for clipping samples as done on Line 16

Remark 4.4.2. *While simulated annealing is usually run in parallel with random initialization to converge to multiple local optima, setting up the neighbourhood function in such a way indicates that the algorithm must be run sequentially instead (due to* solution_arr *being passed in* nbhd_configs*).*

Since the hybrid GP model used is not being updated in a streaming manner online, this procedure can be done offline and seems to be fairly efficient in the 1-D case and so it is assumed that it would generalize well to bounded 2/3-D workspaces.

The relevance of introducing a redundancy radius parameter is motivated by (4.17) and Remark 4.4.1. Including it affords working with a greater variety of solutions across the input space and being able to prune them after the fact as it is seen fit to do so depending on how they are ranked by their corresponding energy values. Once the local optima across G_{in} have been found, it would be beneficial to find other suboptimal solutions outside an ϵ ball around them. This corresponds to an increase in the value of p in (4.17) i.e., an increase the degree of freedom in a nominal MPC controller trying to track an arbitrary point in this set.

As far as the clipping is concerned, it is mainly included in case it is necessary to find the points of maximum similarity (minimum discrepancy) between the distributions in $\mathcal{N}_{\text{set}}$. When moving far away from the regions where data points exist, GPs will settle to a constant mean, high variance output. This causes the minimum discrepancy optimization

Algorithm 4 Custom Neighbourhood Function.

```
Require: $x_{\text{current}}$: Initialization point
Require: $\sigma$: Step size
Require: solution_arr: Current array of accumulated solutions.
Require: redundancy_radius: Radius around samples in solution_arr where $x_{\text{new}}$ is discarded.
Require: $G_{\text{in}}$: GP input bounds.
 1: procedure NeighbourhoodFunction
 2:   count ← 0
 3:   curr_soln_len ← len(solution_arr)
 4:   while true do
 5:     offset $\sim \mathcal{N}(0, \sigma)$
 6:     $x_{\text{new}} \leftarrow x_{\text{current}} +$ offset
 7:     distances ← $\{\|x_{\text{current}} - x_p^{\text{soln}}\| \ \forall\, p \in \{1, \ldots,$ curr_soln_len$\}$
 8:     if (distances[i] > redundancy_radius $\forall\, p \in \{1, \ldots,$ curr_soln_len$\}$) then
 9:       break
10:     else
11:       count ← count +1
12:       if count = 1000 then
13:         return $x_{\text{current}}$
14:       if count % 10 then
  // Increase search radius if unable to find point that satisfies distance criteria.
15:         $\sigma \leftarrow \sigma * 1.1$
16:   $x_{\text{new}} \leftarrow$ ClipToBounds($x_{\text{new}}, G_{\text{in}}$)
17:   return $x_{\text{new}}$
```

to converge to points very far away from the set G_{in} necessitating an additional step of clipping the considered point to within the bounds of G_{in}.

The proposed optimization algorithm. The final proposed optimization that calls the functions described so far is summarized in Algorithm 5. The key points of note are,

- Initial solutions are randomly selected from the GP input space on Line 2 to allow for a reasonable spread and convergence to different local optima.
- The neighbourhood configuration options and their uses have been elaborated on in prior discussion pertaining to Algorithm 4 (Line 7). A similar discussion has been provided regarding the energy configuration, which in this case is just $\hat{g}$ (Line 5) used

for the objective calculation in (4.19).

- The solution array is grown iteratively over sequential runs and used to inform the redundancy distance in Algorithm 4 (Line 9).
- Finally, SortByEnergy on Line 11 sorts the energy array in ascending order and uses the re-ordered indices to adjust the order of the solution array too. Only the top_k solutions and energies are returned as on Line 12. This is further elaborated on in Section 4.4.1.

Algorithm 5 Sequential simulated annealing.

```
Require: $\hat{g}$: Hybrid GP model
Require: $G_{\text{in}}$: GP input space within which candidates can be sampled.
Require: $T$: Initial temperature
Require: $c\_f$: Cooling factor
Require: $\sigma$: Step size
Require: redundancy_radius: Pruning radius for new candidate samples
Require: $x_{\text{init}}$: Initialization point
Require: $N_{\text{iter}}$: Number of iterations to run for.
Require: top_k: Number of solutions to retain.
 1: procedure SeqSimulatedAnnealing
 2:   initial_solutions ← RandomSample($G_{\text{in}}$, num_runs)
 3:   solution_arr ← $\emptyset$
 4:   energy_arr ← $\emptyset$
    // $\mathcal{N}_{\text{set}}$ distributions generated from $\hat{g}$.
 5:   energy_configs ← $\{\hat{g}\}$
 6:   foreach $k \in$ range(num_runs) do
 7:     nbhd_configs ← {solution_arr, redundancy_radius, $G_{\text{in}}$}
 8:     local_solution, local_energy ← SimulatedAnnealing($T, c\_f, \sigma, x_{\text{init}}$,
        $N_{\text{iter}}$, nbhd_configs, energy_configs)
 9:     solution_arr ← solution_arr ∪ local_solution
10:     energy_arr ← energy_arr ∪ local_energy
11:   sorted_solutions, sorted_energy ← SortByEnergy(solution_arr, energy_arr)
    // Truncate array to top k objective values.
12:   top_k_solutions, top_k_energy ← sorted_solutions[: top_k], sorted_energy[: top_k]
13:   return top_k_solutions, top_k_energy
```

4.4.1 Incorporating optimization results into the nominal controller.

Henceforth, the top_k_solutions array will be denoted as x^{opt} as in (4.17). In order to incorporate x^{opt} into the controller it is first necessary to clarify some points regarding the top_k parameter and explore an alternative weight allocations for (4.17).

Why top k? While a larger value of p in (4.17) would allow for a greater degree of freedom, it is still a tunable hyperparameter. If p is chosen to be large enough that the optimization yields samples in areas of high overlap as a result of exhausting the area of the bounded space (depending on the redundancy radius), then limiting to top k values (or alternatively truncating outputs to within some threshold of the smallest value), yields better results.

Relative weighting scheme for the cost function. An additional solution to the problem described above would be to use an alternate weighting scheme for (4.17) based on the top_k_energy array. This would be done simply by dividing top_k_energy throughout by the first element and then replacing w_p in (4.17) by the weights obtained at the corresponding indices. This is utilized in the experiments.

With access to an array of points where samples collected can be identified as belonging to a particular mode of the hybrid model with relative ease, it is desired to find a way to include this in the optimization that collects samples for region identification. This can be done if one is able to trade-off tracking of the high-level waypoints provided to us with tracking the points in x^{opt}. If the optimization for the nominal system is able to converge reasonably close to the desired waypoints, the added exploration cost term (4.17) should now be able to modify this trajectory in a manner that allows us to collect samples that allow for better region discrimination.

Finally I propose the new cost function for the controller (3.26). First, to obtain a cost comparable to the tracking cost for the high-level waypoints, I set the cost matrix in (4.17) to be $Q_{ref} = \sum_{j=1}^{n_x} Q_{j,j}$ where Q is as used to define the stage cost for the controller (3.26). Then the final cost function is of the form,

$$C_{total,k} = C_{track}(x_k, u_k, x_k^{ref}) + \alpha C_{expl}(x_k, u_k, x^{opt}) \tag{4.21}$$

where C_{track} is the original stage cost function as defined in (3.26). $\alpha \geq 0$ can now be used to trade-off exploration with waypoint tracking. The results for this are described in Section 4.5.1.

4.5 Results

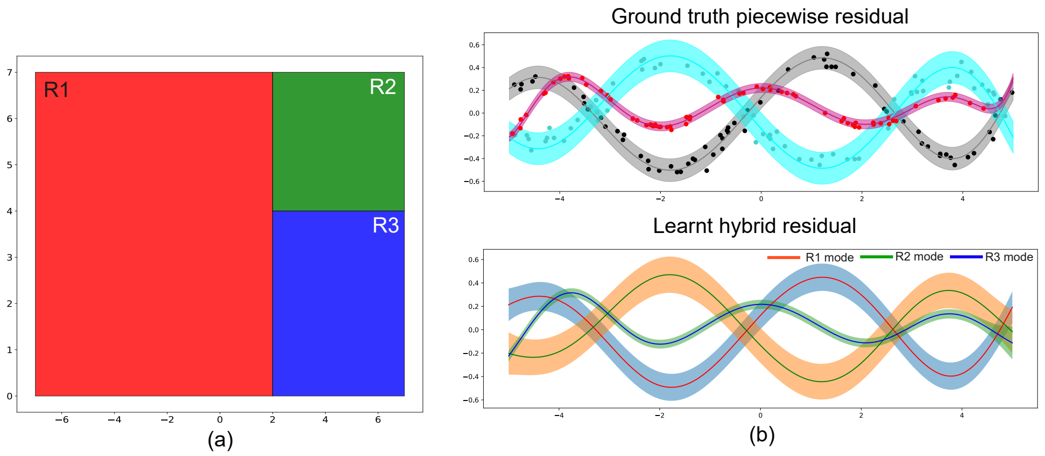


Figure 4.8: Terrain locations and corresponding modes of the piecewise residual dynamics active in each of them.

Simulation Setup The simulation setup is equivalent to that considered in Section 3.7.2 for the most part. The slightly modified residual and regions are shown in Figure 4.8. For the mapping classifier, I use a relatively simple single-layered neural for the experiments. The inputs to the NN are the workspace variables (x, z) and the soft labels are generated by the hybrid GP model as described in Section 4.3.1. The output size is equivalent to the number of terrains considered in this example i.e., 3. The NN starts with a fully-connected layer from the input nodes to a hidden layer containing 32 nodes. The outputs from the hidden layer are passed through a Rectified Linear Unit (ReLU) activation function and then linearly combined to generate $\mathcal{T} = 3$ output logits on which the cross-entropy loss 4.13 is then applied. The Adam [78] optimizer is used for training and backpropagation of gradients. A step learning rate scheduler with an initial learning rate of 0.95 and an exponential decay factor of 0.95 applied every 2 iterations. Due to the discussions on large error bars in Section 4.3.2 as demonstrated by Figure 4.7, all of the tests here involve training 2 NN models on a given dataset and then selecting the one with the lowest loss to minimize the effect of a bad NN mapping classifier initialization on controller performance.

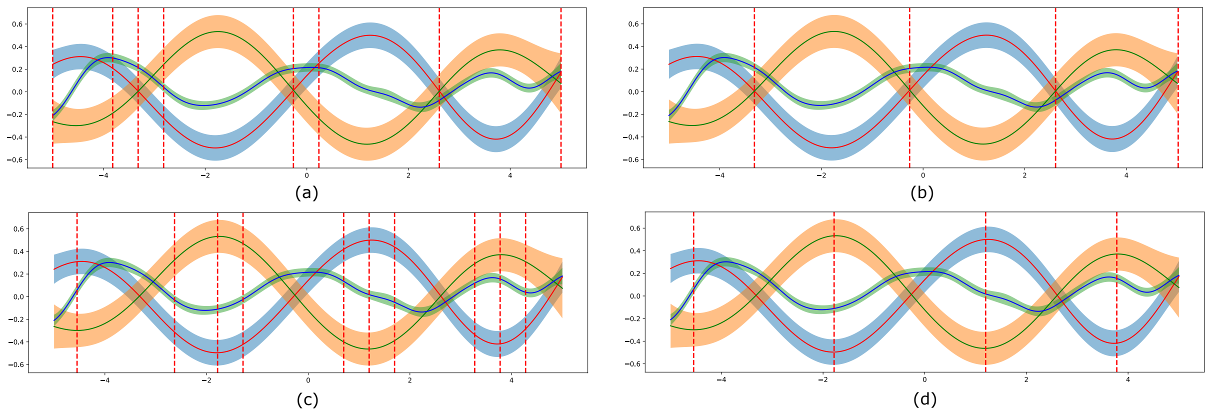


Figure 4.9: (a) / (c) Output from the simulated annealing optimization with num_runs = 20, top_k=10, redundancy_radius=0.45, using a Bhattacharya distance metric that tries to maximize / minimize, respectively, the sum of the pair-wise distances between the elements of $\mathcal{N}_{set}$. (b) / (d) Output from the same optimization but with redundancy radius functionality excluded and solutions within a 0.1 radius of each other being pruned.

4.5.1 Data-efficient mapping results

The results of Algorithm 5 for the Bhattacharya distance metric with and without the redundancy radius parameter are as shown in Figure 4.9. In the case where the redundancy radius parameter is excluded, the solutions obtained in x^{opt} are pruned to prevent plotting of outputs within a 0.1 radius of each other.

Note. This is different to simply setting the redundancy radius parameter to 0.1 as the exclusion causes the candidates for the current iteration of simulated annealing *not* to take into account the positions of existing solutions accumulated in solution_arr.

Figure 4.10 depicts an example of a scaled version of the exploration function and shows how it modifies a tracking cost on $\dot{x} = v_x$ (I consider this variable since $y_k^g = v_x$). This figure serves to provides validation for the inclusions in Algorithm 5 as discussed in Section 4.4.1. With regards to the redundancy radius parameter and the inclusion of top_k results, (a) clearly shows a lower cost over larger parts of the input space when compared with (b) resulting in less bias / restrictions for the optimization.

Now using Figure 4.11 and Table 4.1, I consider how inclusion of this "exploration" cost into the optimization affects the accuracy of the generated dataset and the behaviour of the system. Figure 4.11 depicts a reference trajectory x_k^{ref} to be tracked limited to the

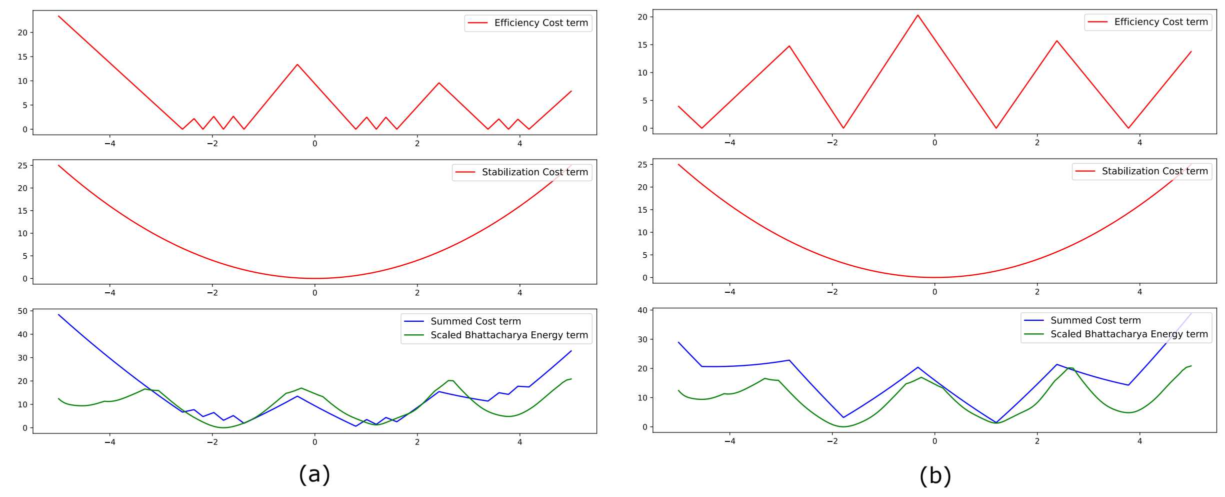
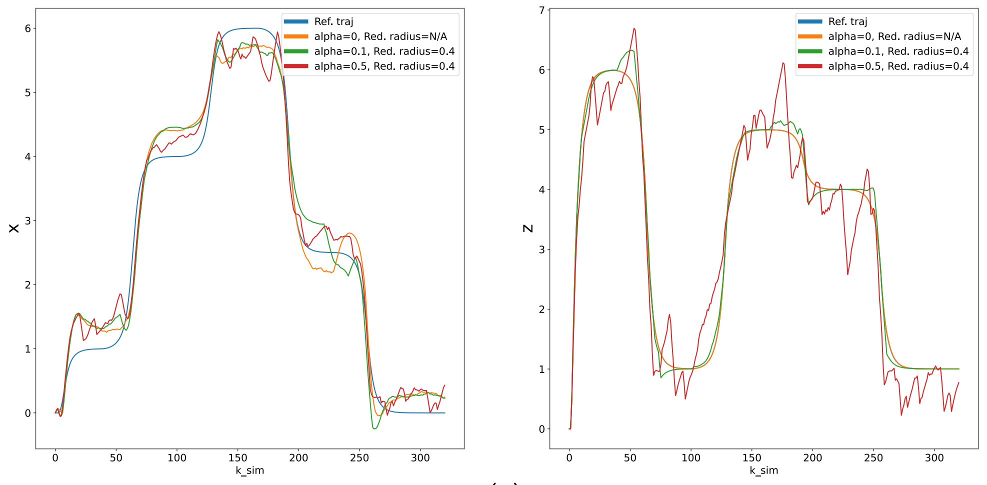


Figure 4.10: A visualization of the cost function that results after incorporating the exploration cost C_{expl} generated using the array x_{opt} produced by finding points of maximum distance between the distributions in $\mathcal{N}_{\text{set}}$. (a) depicts the plot when the redundancy is included and (b) shows the plot when redundancy is not included.

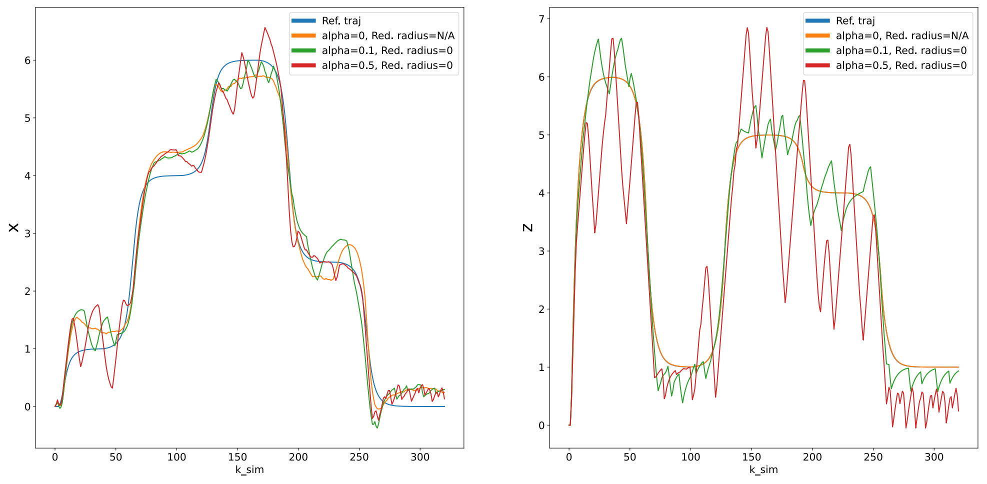
α (4.21)	Red. radius	μ_{ce}	μ_{argmax} (%)
0	-	0.076	93.50
0.1	0	0.063	93.64
0.1	0.4	0.077	93.33
0.5	0	**0.023**	**98.17**
0.5	0.4	0.042	96.4

Table 4.1: A quantitative comparison between the training dataset accuracy generated by various parameters in Algorithm 5. $\alpha = 0$ indicated no exploration and Red. radius $= 0$ indicates a deactivation of the redundancy radius feature. μ_{ce} indicates the average cross-entropy loss per sample computed between the generated soft label and the provided ground truth. μ_{argmax} is the % of soft label that match the ground truth after applying the argmax operator to them.

$x - z$ variables. It also depicts the actual trajectories executed by several variations of the optimization including the exploration cost as indicated in (4.21). The variations include setting α in (4.21) $= \{0.1, 0.5\}$ on the cost function generated when the redundancy radius is included and also when it is excluded.



(a)



(b)

Figure 4.11: A qualitative comparison of the closed-loop trajectories obtained for various assignments parameters in the exploration-based optimization framework. Row (a) shows $x-z$ trajectories when the redundancy radius is excluded and (b) shows trajectories where they are included.

Remarks.

- Neglecting the redundancy radius parameter and setting a high weight for the exploration cost term C_{expl} yields the best training dataset as seen from Table 4.1. However, it also yields significantly erratic behaviour as seen from Figure 4.11 (a).

- When the redundancy radius is included, the cost function is less biasing for the optimization even with a higher weight assigned to C_{expl}. This yields smoother trajectories (as seen in Figure 4.11 (b)) when compared to the plots where the redundancy radius is excluded for the same value of α.

- None of the cases here consider the probability of violation i.e., there is no way to find the optimal value of α that allows exploration while still remaining safe. This is a limitation that I leave to address in future work since it would involve merging the proposed framework with existing safe exploration approaches e.g., [19].

4.5.2 Iterative terrain mapping results

In this section, I demonstrate how I can iteratively train a classifier over multiple runs to obtain an improvement in closed loop performance over time. For this example, the desired repetitive task is defined by the high-level waypoints, $W = [(1,6),(4,1),(6,5),(2.5,3),(0,1)]$. However, to better demonstrate the approach, it is assumed initially that the quadrotor is allowed to track a more restrictive path to obtain samples defined by the waypoints $W_{init} = [(0.5,4),(1,1),(2,3),(1,4)]$. In both cases, the closed-loop trajectories are run for 160 simulation steps and the residual model is the same as shown in Figure 4.8. As indicated in Algorithm 4.3.1, the dataset generated by the current trajectory is concatenated with those generated by previous runs and used to retrain the classifier *from scratch*.

The iterative procedure and results are demonstrated through the qualitative outputs shown in Figure 4.12. The classifier is capable of consistently bringing down the closed-loop cost over multiple runs of the experiment as indicated in Table 4.2.

Run number	μ_{cl}^{c}	+/- errors
1	43.21	8.06/4.46
2	8.11	4.52/0.33

Table 4.2: A comparison of closed-loop performance over repeated runs of the system on the repetitive tracking task. μ_{cl}^{c} denotes the average closed-loop cost for a given run number and the +/- signify the minimum and maximum deviations from the mean computed over 15 simulations.

Remarks.

- On this relatively simple example, the classifier is able to converge to the ground truth over the tracked path within 2 training dataset updates as seen in the last row

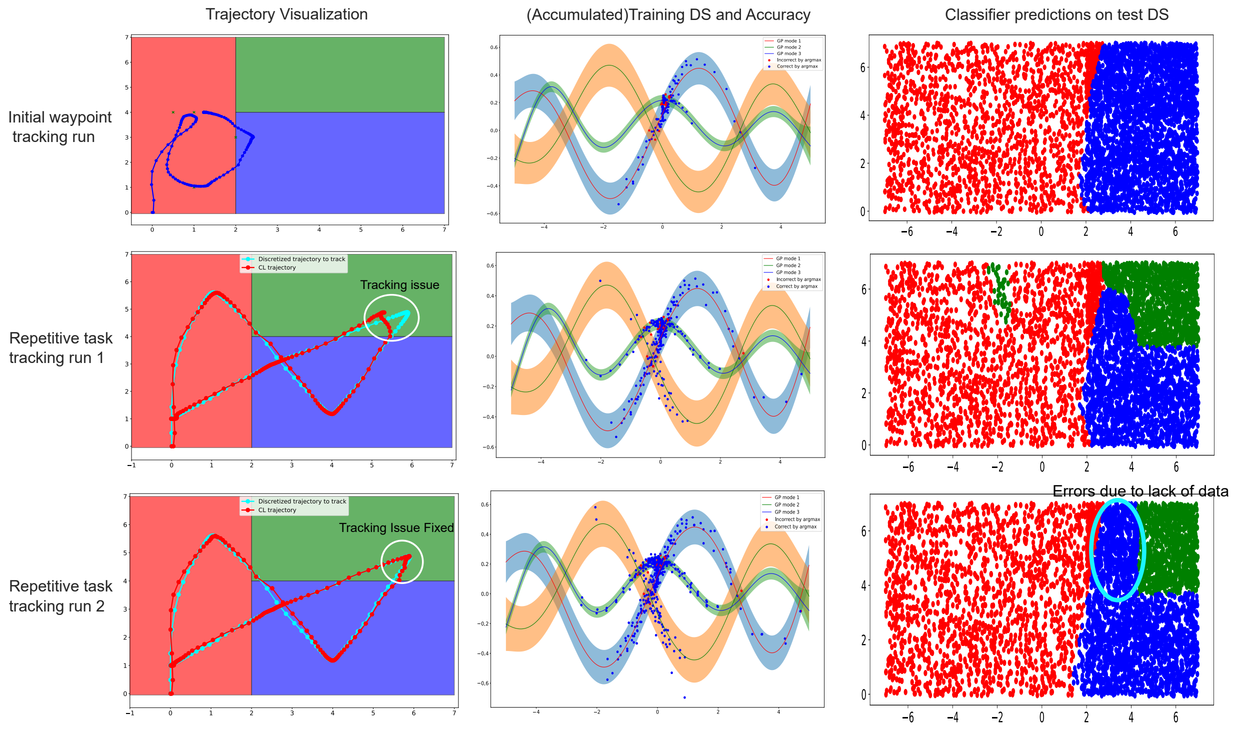


Figure 4.12: Plots demonstrating the training procedure for the iterative mapping-based classifier and the benefits it yields on closed-loop trajectories over repeated runs on the repetitive tracking task.

of Figure 4.12. As a result, runs past 2 are not shown in Table 4.2 since the cost converges to a local optima consistently on and after run 2.

- It is clear that the restricted task defined by W_{init} does not allow for the collection of any samples in R_2 (as shown in green in Figure 4.8). As a result, the first run on the repetitive task demonstrates poor tracking performance on waypoints in this region.
- After collecting samples during the first run, the classifier predictions now reflect R_2 and show significantly improved tracking performance in that region during the second run.
- It is clear that R_2 is not able to be identified completely even after run 2. This is down to the fact that the classifier is liable to incorrect generalize region assignments to parts of the workspace where samples have not been collected.

In terms of notions of safety, I provide a qualitative analysis of results for a task of tracking waypoints close to the boundary. In this task, the majority of the setup remains the same as before except for the waypoint arrays. I choose $W_{init} = [(6,1),(6,6),(1,6),(1,1)]$ and $W = [(6.7,0.3),(6.7,6.7),(0.3,6.7),(0.3,0.3)]$. The qualitative results for the iterative mapping are as shown in Figure 4.13.

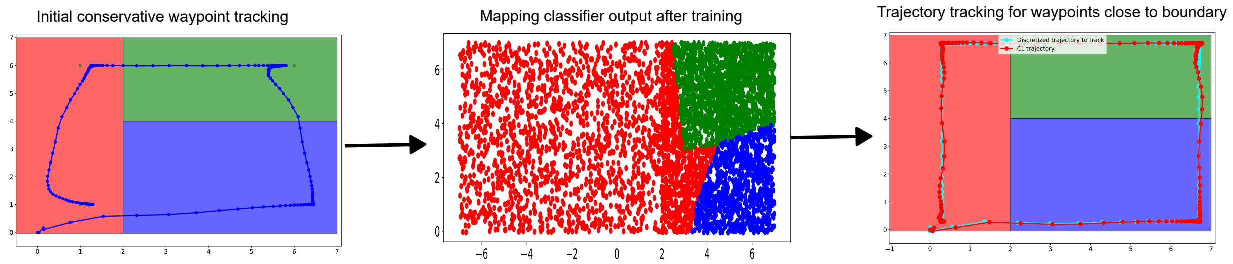


Figure 4.13: Plots demonstrating the possible application of the framework to iterative safe exploration under uncertainty.

Remarks.

- In terms of safety, if the initial waypoint tracking task is relatively conservative and there is some knowledge about how quickly the terrains can vary for a given environment, then the controller appears to perform well on the edge tracking task. This demonstrates potential for the approach to be applied to an iterative learning framework for safe / robust exploration similar to [114, 19] but formalizing this is left for future work.
- Collisions reduce the number of samples available to train on since collision dynamics differ from the residuals and need to be neglected from the training dataset. Incorporating this into the pipeline would thus benefit data efficiency in addition to safety.

4.6 Conclusion

In this Chapter, I proposed a "terrain"-mapping methodology to identify terrain locations across an unknown environment by leveraging an apriori provided hybrid GP model. The use of the soft labels generated by this hybrid model was validated through tests on classifier accuracy. A simulated annealing-based framework was proposed to improve the quality of the dataset available for classifier training. This was demonstrated to be dependent on the inclusion of redundant (sub-optimal) solutions to prevent limiting the optimization due to a dominating exploration cost term.

Mapping tests performed on repetitive tasks were demonstrated to exhibit significant reductions in closed-loop cost over successive iterations as a result of identifying the underlying ground truth region distribution over the trajectory.

In terms of limitations, the controller is not yet capable of handling new environment with Out of Distribution (OOD) terrains i.e., those for which a corresponding mode is not yet present in the hybrid model. Moreover, a formal proof of safety guarantees is hard to present due to the black-box, unexplainable nature of the NN classifier in its current form. These limitations are further elaborated on in Section 5.1.2.

Chapter 5

Conclusion

In this thesis, I have investigated control solutions to chance-constrained optimal control problems concerning systems with piece-wise residual dynamics. Initially, I considered the problem of optimizing over the space of trajectories defined in part by discrete variables resulting in an MINLP. The developed hybrid models were demonstrated to have significant modelling advantages over the use of a single model that has been prevalent in literature so far. Through simulation, the resulting controller that incorporated this model was shown not to suffer from both over-conservatism and under-conservatism in cases where a baseline single GP controller did.

An issue with this controller is the high computational complexity that results from the incorporation of discrete optimization variables making application to most practical systems, requiring relatively long open-loop horizons, intractable. The solution devised for this problem involved using a hierarchical planner-controller architecture to generate reference trajectories that could then parametrize the discrete variables for the controller. This, combined with other approximations, reduced the problem to an NLP with solve times comparable to that of nominal MPC while trading off retaining the benefits of the previously proposed MINLP controller with computational tractability.

I concluded the investigation by relaxing a significant assumption made in the previous two problems viz. the polytopic constraints that define the partitioning regions are known. The solution involves computing the likelihood with which a measured residual term was drawn from each of a set of probabilistic models and then training a classifier to approximate the ground truth terrain mapping function. The approach, demonstrated via simulation on a repetitive task in an environment where the distribution of terrains is unknown, shows significant reduction in closed-loop cost over time. An additional module is also proposed to improve the efficiency with which samples can conclusively be linked to being generated by a specific mode of the hybrid model.

5.1 Limitations and Future Research Directions

Over the course of working on this thesis, several avenues were left unexplored and are detailed here to represent future research directions or improvements to the modules proposed.

5.1.1 MINLP to NLP conversion.

Recovering optimality benefits of an MINLP. One of the issues with the NLP version of the hybrid model-based controller proposed in Section 3.6 is that there is no longer a search over the space of trajectories (and hence assignments to δ) but rather just a fixed trajectory to track. As a result, some of the optimality benefits that stem from the MINLP are lost. To recover this, randomized planning approaches like RRT* (as described in Section 3.6.3) can generate paths that are quite different from each other and hence might constitute varying assignments to the δ array. Attempting to then track each generated spline-interpolated trajectory using parallel instances of the parametrized controller and selecting the best one would help us maintain some notion of optimizing over the space of discrete variables. This might be necessary in more complex scenarios like those involving dynamic (or even static) obstacle avoidance.

Re-planning conditions. Algorithm 1 generates a lot of quantities that are local to the trajectory being tracked by the online NLP controller proposed in Section 3.6.2. When there is a deviation from the reference trajectory in closed-loop, these apriori computed quantities might no longer be valid. Setting up conditions under which re-planning must occur in order to maintain validity of these trajectory-centric quantities would be essential in practical scenarios. The use of anytime RRT* variants [75] might still allow for real-time implementation of the controller in (3.26) if re-planning is to be included.

Incorporating uncertainty estimates into the pipeline. (3.26) does away with including the variance costs in the controller to prevent online computation of the covariance matrices in the optimization. When Assumption 3.2.4 holds, this might not be particularly limiting. However, when this is not the case, a large uncertainty in the generated plan implies that the δ_k's generated based on the reference trajectory might not be valid.

Incorporating the uncertainty estimates generated by the hybrid model into the heuristic/objective function for the offline planner to steer it away from high uncertainty regions of the GP input space that result from lack of data might help especially if the uncertainty ends up compounding over the horizon otherwise. Alternatively, something similar would also be required if it is desired to explore areas of the GP input space with low data density.

5.1.2 Terrain mapping-based control.

Improving safety guarantees. One of the key limitations of the approach proposed in Section 4.3 is the lack of guarantees provided on safety. The factors that contribute to this and some implications are highlighted here.

- With black-box models like NNs, it can be hard to guarantee convergence of the mapping classifiers to the ground truth even with a reasonably large dataset. As a result, proving the convergence of $\hat{T}(x) \to T(x)$, desired in Problem 4 to provide safety guarantees, is hard when using such a model.

- Section 4.3.2 demonstrates benefits to training classifiers using soft labels. However, when incorporating this classifier into the controller, it does not take into account the output logits themselves but rather computes an argmax over them to assign values to the reference δ array. This is a limitation which becomes apparent when considering a normalized logit output prediction of $[0.33, 0.33, 0.34]^T$. Despite the high uncertainty, an argmax assigns $\delta_{3,k} = 1$ where the residual could be wildly different from those in the other two regions. Instead, it would be beneficial to use a fuzzy logic-based control approach to leverage the logits in the predictions generated by the classifier. Being able to combine the information from multiple modes based on these soft predictions could benefit not only safety but also performance.

- Another issue, similar to the above, is when the classifier is overconfident of its prediction. This is a problem when the prediction is in a region of low data density. For example, if the waypoints defining the repetitive task has changed but the old model is still utilized, its predictions on inputs that are reasonably far away from previously collected data points must be treated a degree of skepticism. In such cases, it would be beneficial to quantify some notion of worst case probabilistic shrunk sets and adhere to this when such a situation arises.

- The results in Section 4.5.1 highlights issues with safe data collection while still obtaining data samples that allow us to easily identify the active mode at that point with confidence. Similar to the above, worst case probabilistic shrunk sets might help initially provide safety benefits during the process of data collection. To reduce over-conservative behaviours, these worst case sets can be iteratively refined over time as more data is identified and the environment is better understood.

Extensions to a continually changing setting. The benefits of the proposed mapping approach rely on the fact that the task to be carried out is repetitive and the data samples collected from previous trajectory can be utilized to obtain more accurate predictions in an MPC scheme. However, when moving to a setting where the environment has the potential to change continually (e.g., an autonomous vehicle on a highway instead of a wheeled robot in a closed warehouse) there could be potential issues with using the classifier to predict assignments to discrete variables. Using a hybrid control scheme with adaptive and MPC control in parallel could be beneficial in such settings. Moreover, MPC predictions could be better informed if a sensor fusion approach was adopted using camera data to help with terrain identification.

Adapting to region changes over time. The underlying model active across different parts of the workspace is subject to change over time. A limitation of the proposed classifier is that it uses previous data samples to re-train from scratch in between runs. In contrast, it would be better to incorporate the priors generated by previous iterations of the classifier into (4.6) while also prioritizing the likelihood samples generated if a region and has not been visited for some time. This would allow the classifier to be trained iteratively allowing for adaptation as opposed to concatenating all the datasets together as done in Algorithm 4.3.1.

5.1.3 Online learning and control.

Guarantees on stability and recursive feasibility. The proposed controllers currently do not provide any guarantees in terms of stability and recursive feasibility. There is some work that tries to address such issues as highlighted in [86]. It would be relevant to see if existing results can be leveraged, or the necessary theory can be developed, to endow the controllers with these crucial additional properties. This would allow for a more comprehensive theoretical analysis of the performance of these controllers.

Building hybrid models in an unsupervised manner. A limitation of this thesis was that it was assumed that the dataset provided to train the hybrid GP residual had labels that indicated which mode of the residual a sample was drawn from. While there has been worked that tries to address this (as elaborated on in Section 1.1.1) it would be relevant from a practical standpoint to see whether these approaches mesh with the mapping framework proposed in Chapter 4. This would allow for the extension of the proposed approach to environments which contain terrains for which datasets have not yet been provided (corresponding to a relaxation of Assumption 4.2.1) leading to the absence of a mode corresponding to that terrain in the hybrid model.

Sparse GP approximations for hybrid models. Often sparse approximations are computed using the previous open-loop optimization solution as inducing points [55, 73, 61]. However, when dealing with region based information, there might not have enough points across all regions for each model to be sufficiently approximated. A question to be investigated is whether this matters to performance and safety as long as approximation is good enough about the points in the future horizon. In the case of simple tasks this might be sufficient but in complex environments with other dynamic agents this could no longer be the case.

5.1.4 Real-World Implementation

There was insufficient time to implement these techniques on a real-world platform. In addition to the practical considerations highlighted across remarks in Chapter 3 and 4, it remains to be seen whether the residuals encountered in practice follow the same trend as has been considered in the illustrative examples shown in this thesis. For example, tyre friction models have areas of high overlap where the dynamics are quite similar outside of which the models can deviate significantly. This is an intermediate between the two cases considered in 3.4 and ways of efficiently modelling and incorporating such residuals into the control formulation remain to be explored.

References

[1] Stochastic linear model predictive control with chance constraints – a review. *Journal of Process Control*, 44:53–67, 2016.

[2] Hirotugu Akaike. Fitting autoregreesive models for prediction. In *Selected Papers of Hirotugu Akaike*, pages 131–135. Springer, 1969.

[3] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[4] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[5] Patrick R Amestoy, Iain S Duff, and J-Y L'excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer methods in applied mechanics and engineering*, 184(2-4):501–520, 2000.

[6] Brian DO Anderson and Arvin Dehghani. Challenges of adaptive control–past, permanent and future. *Annual reviews in control*, 32(2):123–135, 2008.

[7] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2019.

[8] Shan Ba and V. Roshan Joseph. Composite gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, 6(4), dec 2012.

[9] François Bachoc, Fabrice Gamboa, Jean-Michel Loubes, and Nil Venet. A gaussian process regression model for distribution inputs. *IEEE Transactions on Information Theory*, 64(10):6620–6637, 2017.

[10] Mohammed Saad Faizan Bangi and Joseph Sang-Il Kwon. Deep hybrid model-based predictive control with guarantees on domain of applicability. *AIChE Journal*, 69(5):e18012, 2023.

[11] Yajie Bao, Kimberly J Chan, Ali Mesbah, and Javad Mohammadpour Velni. Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust bayesian neural networks. *International Journal of Robust and Nonlinear Control*, 33(5):3312–3333, 2023.

[12] Nicholas H. Barbara, Max Revay, Ruigang Wang, Jing Cheng, and Ian R. Manchester. Robustneuralnetworks.jl: a package for machine learning and data-driven control with certified robustness, 2023.

[13] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza. Neurobem: Hybrid aerodynamic quadrotor model. *arXiv preprint arXiv:2106.08015*, 2021.

[14] Alberto Bemporad. A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 2022.

[15] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[16] Alberto Bemporad, Carlo A Pascucci, and Claudio Rocchi. Hierarchical and hybrid model predictive control of quadcopter air vehicles. *IFAC Proceedings Volumes*, 42(17):14–19, 2009.

[17] Mouhacine Benosman. Model-based vs data-driven adaptive control: an overview. *International Journal of Adaptive Control and Signal Processing*, 32(5):753–776, 2018.

[18] Julian Berberich, Johannes Köhler, Matthias A Müller, and Frank Allgöwer. Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, 66(4):1702–1717, 2020.

[19] Felix Berkenkamp and Angela P. Schoellig. Safe and robust learning control with Gaussian processes. In *2015 European Control Conference (ECC)*, pages 2496–2501, Linz, Austria, July 2015. IEEE.

[20] Karl Berntorp and Kitano Hiroaki. Bayesian learning of tire friction with automotive-grade sensors by gaussian-process state-space models. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6681–6686, 2019.

[21] Mickaël Binois and Robert Gramacy. hetgp : Heteroskedastic gaussian process modeling and sequential design in r. *Journal of Statistical Software*, 98, 07 2021.

[22] Lars Blackmore. A probabilistic particle control approach to optimal, robust predictive control. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6240, 2006.

[23] Lars Blackmore, Masahiro Ono, and Brian Charles Williams. Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27:1080–1094, 2011.

[24] Henk AP Blom, John Lygeros, M Everdij, S Loizou, and K Kyriakopoulos. *Stochastic hybrid systems: theory and safety critical applications*, volume 337. Springer, 2006.

[25] Pierre Bonami and Jon Lee. Bonmin user's manual.

[26] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[27] Mathieu Bourdeau, Xiao qiang Zhai, Elyes Nefzaoui, Xiaofeng Guo, and Patrice Chatellier. Modeling and forecasting building energy consumption: A review of data-driven techniques. *Sustainable Cities and Society*, 48:101533, 2019.

[28] Eric Bradford, Lars Imsland, Dongda Zhang, and Ehecatl Antonio del Rio Chanona. Stochastic data-driven model predictive control using gaussian processes. *Computers & Chemical Engineering*, August 2020.

[29] Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning, 2021.

[30] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[31] Thang D Bui, Cuong Nguyen, and Richard E Turner. Streaming Sparse Gaussian Process Approximations. page 9.

[32] Martin Buss, Markus Glocker, Michael Hardt, Oskar Von Stryk, Roland Bulirsch, and Günther Schmidt. Nonlinear hybrid dynamical systems: modeling, optimal control, and applications. In *Modelling, Analysis, and Design of Hybrid Systems*, pages 311–335. Springer, 2002.

[33] Erion Bwambale, Felix K Abagale, and Geophrey K Anornu. Data-driven model predictive control for precision irrigation management. *Smart Agricultural Technology*, 3:100074, 2023.

[34] Roberto Calandra, Andre Seyfarth, Jan Peters, and Marc Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 76, 06 2015.

[35] E.F. Camacho, D.R. Ramirez, D. Limon, D. Muñoz de la Peña, and T. Alamo. Model predictive control techniques for hybrid systems. *Annual Reviews in Control*, 34(1):21–31, April 2010.

[36] Andrea Carron, Elena Arcari, Martin Wermelinger, Lukas Hewing, Marco Hutter, and Melanie N Zeilinger. Data-driven model predictive control for trajectory tracking with a robotic arm. *IEEE Robotics and Automation Letters*, 4(4):3758–3765, 2019.

[37] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

[38] Girish Chowdhary, Hassan A. Kingravi, Jonathan P. How, and Patricio A. Vela. Bayesian nonparametric adaptive control using gaussian processes. *IEEE Transactions on Neural Networks and Learning Systems*, 26(3):537–550, 2015.

[39] Fabian Christ, Alexander Wischnewski, Alexander Heilmeier, and Boris Lohmann. Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients. *Vehicle System Dynamics*, 59(4):588–612, 2021.

[40] Lehel Csato. Gaussian Processes - Iterative Sparse Approximations. page 117.

[41] Lehel Csató. Gaussian processes - iterative sparse approximations. 03 2002.

[42] Olaf Czogalla, Robert Hoyer, and Ulrich Jumar. Modelling and simulation of controlled road traffic. In *Modelling, Analysis, and Design of Hybrid Systems*, pages 419–435. Springer, 2002.

[43] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

[44] Manoj Kumar Debnath, Tarakanta Jena, and Smaran Kumar Sanyal. Frequency control analysis with pid-fuzzy-pid hybrid controller tuned by modified gwo technique. *International Transactions on Electrical Energy Systems*, 29(10):e12074, 2019.

[45] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

[46] Tommaso Dreossi, Alexandre Donzé, and Sanjit A. Seshia. Compositional falsification of cyber-physical systems with machine learning components, 2017.

[47] David Duvenaud. *Automatic model construction with Gaussian processes.* PhD thesis, University of Cambridge, 2014.

[48] David Duvenaud. *Automatic model construction with Gaussian processes.* PhD thesis, 11 2014.

[49] David D. Fan, Jennifer Nguyen, Rohan Thakker, Nikhilesh Alatur, Ali-akbar Agha-mohammadi, and Evangelos A. Theodorou. Bayesian learning-based adaptive control for safety critical systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4093–4099, 2020.

[50] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[51] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 2997–3004. IEEE, 2014.

[52] G. Gao, Hao Jiang, Jeroen Vink, Chaohui Chen, Yaakoub el Khamra, and J. Ita. Gaussian mixture model fitting method for uncertainty quantification by conditioning to production data. *Computational Geosciences*, 24:663–681, 04 2020.

[53] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration, 2018.

[54] Jorge L Garriga and Masoud Soroush. Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8):3505–3515, 2010.

[55] Agathe Girard, Carl Edward Rasmussen, Joaquin Quiñonero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, page 545–552, Cambridge, MA, USA, 2002. MIT Press.

[56] Lars Grüne, Jürgen Pannek, Lars Grüne, and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2017.

[57] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[58] Wei He, Zhao Yin, and Changyin Sun. Adaptive neural network control of a marine vessel with constraints using the asymmetric barrier lyapunov function. *IEEE transactions on cybernetics*, 47(7):1641–1651, 2016.

[59] Wei He, Zhao Yin, and Changyin Sun. Adaptive neural network control of a marine vessel with constraints using the asymmetric barrier lyapunov function. *IEEE transactions on cybernetics*, 47(7):1641–1651, 2016.

[60] Lukas Hewing, Andrea Carron, Kim P. Wabersich, and Melanie N. Zeilinger. On a correspondence between probabilistic and robust invariant sets for linear systems. In *2018 European Control Conference (ECC)*, pages 1642–1647, 2018.

[61] Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, nov 2020.

[62] Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Cautious NMPC with gaussian process dynamics for autonomous miniature race cars. In *2018 European Control Conference (ECC)*. IEEE, jun 2018.

[63] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[64] Lukas Hewing and Melanie N. Zeilinger. Performance analysis of stochastic model predictive control with direct and indirect feedback. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 672–678, 2020.

[65] Lukas Hewing and Melanie N. Zeilinger. Scenario-based probabilistic reachable sets for recursively feasible stochastic model predictive control. *IEEE Control Systems Letters*, 4(2):450–455, 2020.

[66] Lukas Hewing and Melanie Nicole Zeilinger. Indirect and direct feedback in stochastic model predictive control. 2020.

[67] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[68] Zhongsheng Hou, Ronghu Chi, and Huijun Gao. An overview of dynamic-linearization-based data-driven control and applications. *IEEE Transactions on Industrial Electronics*, 64(5):4076–4090, 2016.

[69] Naira Hovakimyan and Chengyu Cao. *1 adaptive control theory: Guaranteed robustness with fast adaptation.* SIAM, 2010.

[70] Naira Hovakimyan, Flavio Nardi, Anthony Calise, and Nakwan Kim. Adaptive output feedback control of uncertain nonlinear systems using single-hidden-layer neural networks. *IEEE Transactions on neural networks*, 13(6):1420–1431, 2002.

[71] Marvin Jung, Paulo Renato da Costa Mendes, Magnus Önnheim, and Emil Gustavsson. Model predictive control when utilizing lstm as dynamic models. *Engineering Applications of Artificial Intelligence*, 123:106226, 2023.

[72] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[73] Sanket Kamthe and Marc Peter Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control, 2017.

[74] Nikolas Kantas, JM Maciejowski, and A Lecchini-Visintini. Sequential monte carlo for model predictive control. *Nonlinear model predictive control: Towards new challenging applications*, pages 263–273, 2009.

[75] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, 2011.

[76] Robert E. Kass, Bradley P. Carlin, Andrew Gelman, and Radford M. Neal. Markov chain monte carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100, 1998.

[77] Saeed Khankalantary, Pouya Badri, and Hassan Mohammadkhani. Designing a hierarchical model-predictive controller for tracking an unknown ground moving target using a 6-dof quad-rotor. *International Journal of Dynamics and Control*, 9:985–999, 2021.

[78] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[79] Kevin Kircher. Robust and stochastic optimization notes, Fall 2015.

[80] Bahare Kiumarsi, Kyriakos G Vamvoudakis, Hamidreza Modares, and Frank L Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, 29(6):2042–2062, 2017.

[81] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, Joschka Boedecker, and Andreas Krause. Learning-based model predictive control for safe exploration and reinforcement learning, 2019.

[82] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[83] Eugene Lavretsky. Adaptive control: Introduction, overview, and applications. https://www.cds.caltech.edu/archive/help/uploads/wiki/files/140/IEEE_WorkShop_Slides_Lavretsky.pdf. Accessed: 1-7-2023.

[84] Quoc V Le, Alex J Smola, and Stéphane Canu. Heteroscedastic gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, pages 489–496, 2005.

[85] Frank L Lewis, Aydin Yesildirek, and Kai Liu. Multilayer neural-net robot controller with guaranteed tracking performance. *IEEE Transactions on neural networks*, 7(2):388–399, 1996.

[86] Michael Maiworm, Daniel Limon, and Rolf Findeisen. Online learning-based model predictive control with gaussian process models and stability guarantees. *International Journal of Robust and Nonlinear Control*, 31(18):8785–8812, 2021.

[87] D Mariano-Hernández, L Hernández-Callejo, A Zorita-Lamadrid, O Duque-Pérez, and F Santos García. A review of strategies for building energy management system: Model predictive control, demand side management, optimization, and fault detect & diagnosis. *Journal of Building Engineering*, 33:101692, 2021.

[88] Albert W Marshall, Ingram Olkin, and Barry C Arnold. Inequalities: theory of majorization and its applications. 1979.

[89] Andrew Mchutchon and Carl Rasmussen. Gaussian process training with input noise. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[90] Christopher D. McKinnon and Angela P. Schoellig. Learning multimodal models for robot dynamics online with a mixture of gaussian process experts. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 322–328, 2017.

[91] Christopher D McKinnon and Angela P Schoellig. Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks. *IEEE Robotics and Automation Letters*, 4(2):2180–2187, 2019.

[92] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[93] Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(95):2651–2667, 2006.

[94] Ioanna Mitsioni, Yiannis Karayiannidis, Johannes A Stork, and Danica Kragic. Data-driven model predictive control for the contact-rich task of food cutting. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 244–250. IEEE, 2019.

[95] Pieter J Mosterman, Manuel A Pereira Remelhe, Sebastian Engell, and Martin Otter. Simulation for analysis of aircraft elevator feedback and redundancy control. In *Modelling, analysis, and design of hybrid systems*, pages 369–390. Springer, 2002.

[96] Tomáš Nagy, Ahmad Amine, Truong X Nghiem, Ugo Rosolia, Zirui Zang, and Rahul Mangharam. Ensemble gaussian processes for adaptive autonomous driving on multi-friction surfaces. *arXiv preprint arXiv:2303.13694*, 2023.

[97] Radford M Neal. *Probabilistic inference using Markov chain Monte Carlo methods.* 1993.

[98] Masahiro Ono and Brian C. Williams. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint.

[99] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[100] Chris J. Ostafew, Angela P. Schoellig, and Timothy D. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4029–4036, May 2014. ISSN: 1050-4729.

[101] Chris J. Ostafew, Angela P. Schoellig, and Timothy D. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4029–4036, 2014.

[102] Hans B. Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle System Dynamics*, 21(sup001):1–18, 1992.

[103] Joel A Paulson, Edward A Buehler, Richard D Braatz, and Ali Mesbah. Stochastic model predictive control with joint chance constraints. *International Journal of Control*, 93(1):126–139, 2020.

[104] Ian R Petersen and Roberto Tempo. Robust control of uncertain systems: Classical results and recent developments. *Automatica*, 50(5):1315–1335, 2014.

[105] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, dec 2020.

[106] Alexander S Poznyak, Edgar N Sanchez, and Wen Yu. *Differential neural networks for robust nonlinear control: identification, state estimation and trajectory tracking.* World Scientific, 2001.

[107] Krupa Prag, Matthew Woolway, and Turgay Celik. Toward data-driven optimal control: A systematic review of the landscape. *IEEE Access*, 10:32190–32212, 2022.

[108] Christopher J. Pratt and Kam K. Leang. Dynamic underactuated flying-walking (duck) robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3267–3274, 2016.

[109] Ananth Ranganathan, Ming-Hsuan Yang, and Jeffrey Ho. Online Sparse Gaussian Process Regression and Its Applications. *IEEE Transactions on Image Processing*, 20(2):391–404, February 2011.

[110] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[111] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

[112] Beibei Ren, Shuzhi Sam Ge, Keng Peng Tee, and Tong Heng Lee. Adaptive neural control for output feedback nonlinear systems using a barrier lyapunov function. *IEEE Transactions on Neural Networks*, 21(8):1339–1345, 2010.

[113] Garrett Robert Rose. Numerical methods for solving optimal control problems. Master's thesis, University of Tennessee, 2015.

[114] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework, 2016.

[115] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

[116] Robert M Sanner and Jean-Jacques E Slotine. Gaussian networks for direct adaptive control. In *1991 American control conference*, pages 2153–2159. IEEE, 1991.

[117] Alan D. Saul, James Hensman, Aki Vehtari, and Neil D. Lawrence. Chained gaussian processes, 2016.

[118] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[119] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5-6):1327–1349, 2021.

[120] Science and Technology Facilities Council (STFC). Coin-hsl solvers. https://licences.stfc.ac.uk/product/coin-hsl. Accessed: 5-7-2023.

[121] Yong Shi, Wei Dai, Wen Long, and Bo Li. Deep kernel gaussian process based financial market predictions, 2021.

[122] Francesco Smarra, Achin Jain, Tullio de Rubeis, Dario Ambrosini, Alessandro D'Innocenzo, and Rahul Mangharam. Data-driven model predictive control using random forests for building energy optimization and climate control. *Applied Energy*, 226:1252–1272, 2018.

[123] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. 01 2005.

[124] Marion Sobotka. Hybrid dynamical system methods for legged robot locomotion with variable ground contact. 2007.

[125] Nilesh Suriyarachchi, Rien Quirynen, John S Baras, and Stefano Di Cairano. Optimization-based coordination and control of traffic lights and mixed traffic in multi-intersection environments. In *2023 American Control Conference (ACC)*, pages 3162–3168. IEEE, 2023.

[126] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots, 2018.

[127] Russ Tedrake. *Underactuated Robotics*. 2022.

[128] C.J. Tomlin, I. Mitchell, A.M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.

[129] Guillem Torrente, Elia Kaufmann, Philipp Fohn, and Davide Scaramuzza. Data-Driven MPC for Quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, April 2021.

[130] Arjan J Van Der Schaft and Hans Schumacher. *An introduction to hybrid dynamical systems*, volume 251. springer, 2007.

[131] Dennis Harald Van Hessem. Stochastic inequality constrained closed-loop model predictive control with application to chemical process operation. 2004.

[132] Bart PG Van Parys, Daniel Kuhn, Paul J Goulart, and Manfred Morari. Distributionally robust control of constrained stochastic systems. *IEEE Transactions on Automatic Control*, 61(2):430–442, 2015.

[133] Andrea Lecchini Visintini, William Glover, John Lygeros, and Jan Maciejowski. Monte carlo optimization for conflict resolution in air traffic control. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):470–482, 2006.

[134] Vivek Vittaldev, Ryan P Russell, and Richard Linares. Spacecraft uncertainty propagation using gaussian mixture models and polynomial chaos expansions. *Journal of Guidance, Control, and Dynamics*, 39(12):2615–2626, 2016.

[135] RICHARD VON MISES. Chapter vii - probability inference. bayes' method. In RICHARD VON MISES, editor, *Mathematical Theory of Probability and Statistics*, pages 329–367. Academic Press, 1964.

[136] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[137] Chunyi Wang and Radford M. Neal. Gaussian process regression with heteroscedastic or non-gaussian residuals, 2012.

[138] Ruigang Wang and Ian Manchester. Direct parameterization of lipschitz-bounded deep networks. In *International Conference on Machine Learning*, pages 36093–36110. PMLR, 2023.

[139] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010.

[140] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[141] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015.

[142] Wee Chin Wong, Ewan Chee, Jiali Li, and Xiaonan Wang. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6(11):242, 2018.

[143] Xiao Wu, Jiong Shen, Yiguo Li, and Kwang Y Lee. Data-driven modeling and predictive control for boiler–turbine unit using fuzzy clustering and subspace methods. *ISA transactions*, 53(3):699–708, 2014.

[144] Zheng Yan and Jun Wang. Model predictive control for tracking of underactuated vessels based on recurrent neural networks. *IEEE Journal of Oceanic Engineering*, 37(4):717–726, 2012.

[145] Zheng Yan and Jun Wang. Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. *IEEE transactions on neural networks and learning systems*, 25(3):457–469, 2013.

[146] Zhaocong Yuan, Adam W. Hall, Siqi Zhou, Lukas Brunke, Melissa Greeff, Jacopo Panerati, and Angela P. Schoellig. safe-control-gym: a unified benchmark suite for safe learning-based control and reinforcement learning in robotics, 2022.

[147] Andrea Zanelli, Alexander Domahidi, Juan Jerez, and Manfred Morari. Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, 2020.

[148] Jiu-sun Zeng, Chuan-hou Gao, and Hong-ye Su. Data-driven predictive control for blast furnace ironmaking process. *Computers & Chemical Engineering*, 34(11):1854–1862, 2010.

[149] Dan Zhang and Bin Wei. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control*, 43:188–198, 2017.

[150] Yongchang Zhang, Jialin Jin, and Lanlan Huang. Model-free predictive current control of pmsm drives based on extended state observer using ultralocal model. *IEEE Transactions on Industrial Electronics*, 68(2):993–1003, 2021.

[151] Yingzhe Zheng, Cheng Hu, Xiaonan Wang, and Zhe Wu. Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. *Journal of Process Control*, 128:103005, 2023.

APPENDICES

Appendix A

Dynamics model for a 2-D quadrotor.

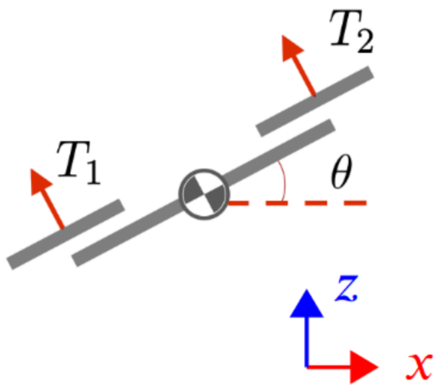


Figure A.1: [146] A 2-D quadrotor model subject to 2 input thrusts T_1, T_2 with tilt angle θ operating in the $x - z$ plane.

The 2-D quadrotor utilized in this thesis is as presented in [146] with all of the parameters in the dynamics and bounds being pulled from either [146] or the associated code-base. The quadrotor operates in the $x - z$ plane with a state and input vector,

$$\mathbf{x} = [x, \dot{x}, z, \dot{z}, \theta, \dot{\theta}]^T \tag{A.1a}$$

$$\mathbf{u} = [u_1, u_2]^T = [T_1, T_2]^T \tag{A.1b}$$

where $(x, z), \theta$ denote the cartesian co-ordinates along the $x - z$ axes and the tilt angle measured CCW from the x-axis respectively. $(\dot{x}, \dot{z}), \dot{\theta}$ represent the velocity along the $x - z$ axes and the tilt angle rate of change respectively.

The equations of motion of this system are of the form,

$$\ddot{x} = \sin\theta\left(u_1 + u_2\right)/m \tag{A.2a}$$

$$\ddot{z} = \cos\theta\left(u_1 + u_2\right)/m - g \tag{A.2b}$$

$$\ddot{\theta} = \frac{\left(u_2 - u_1\right)}{I_{yy}}\frac{l}{\sqrt{2}}, \tag{A.2c}$$

The assignments to the parameters in (A.2) are as follows,

- $m = 0.027$ kg - quadrotor mass
- $l = 0.0397$ m - arm length of the quadrotor
- $I_{yy} = 1.4 \times 10^{-5}$ kg m^2 - moment of inertia about the y-axis

Bounds for x, z are dependent on the environment and as such are specified in the simulation setup in Section 3.7.2. The velocity limits are set to be $-5 \leq \dot{x} \leq 5$ and similarly for $\dot{z}$. For tilt angles, the limits are defined as $-1.484\text{rad.} \leq \theta \leq 1.484\text{rad.}$ with no constraint on the rate of change of tilt. Bounds on the input are as determined by motor coefficients and minimum and maximum Pulse Width Modulation (PWM) limits and are thus defined as $0.05632 \leq u_i \leq 0.29668 \; \forall \, i \in \{1, 2\}$.

Glossary

Chance-Constrained Model Predictive Controller A type of MPC dealing with probabilistic constraints, usually used for systems involving uncertainty modelled as distributions with infinite support (e.g., Gaussians). xvi

combinatorial explosion Refers to a situation where the number of possible combinations or outcomes of a problem grows exponentially as the problem size increases. This can quickly become overwhelming for computer algorithms, as the time and resources needed to explore every possible combination become impractical, or infeasible necessitating the use of relaxations and/or approximations to the original problem. 38

hybrid This shares the definition of a piecewise system but in the context of this thesis I will use this term to indicate the learnt approximation of some underlying piecewise residual dynamics or to characterize an MPC controller making use of such a model in the dynamics constraint equations. iii, x, 2, 3, 5, 6, 17, 19, 21–25, 27, 29–31, 33–38, 45, 52, 53, 55, 57, 60, 61, 64, 65, 68, 72, 76, 83, 85–88

Mixed Integer Nonlinear Program An extension of non-linear programming where the optimization variables can be continuous or (discrete) integer-valued. These problems can become computationally intractable to solve as the number of variables and constraints grows. xvi

piecewise In the context of this thesis, this term will refer to **true** underlying residuals that vary over different regions of the state-space (or more practically, the workspace of a robotic system). iii, 17, 19, 21, 23, 27, 30, 31, 54–57, 106

polytope A n-dimensional polyhedron often used in MPC to define constraint sets. 13, 20

residual The mismatch between next state of a discrete (or discretized) system predicted by a nominal model derived from first principles and the true measured next state (assuming no measurement error). iii, 2, 9–12, 17, 19, 21–25, 27, 30, 88, 106