

# Real-Time Learning-Based Model Predictive Control: Online Algorithms and Applications in Energy Systems

**Master Thesis**

**Author(s):**

Aoife, Henry

**Publication date:**

2021-03

**Permanent link:**

<https://doi.org/10.3929/ethz-b-000524283>

**Rights / license:**

[In Copyright - Non-Commercial Use Permitted](#)

Aoife Henry

# Real-Time Learning-Based Model Predictive Control: Online Algorithms and Applications in Energy Systems

**Master Thesis**

Automatic Control Laboratory  
Swiss Federal Institute of Technology (ETH) Zurich

Systems & Control  
University of Colorado Boulder

**Supervision**

Prof. Emiliano Dall'Anese, Systems & Control, University of Colorado Boulder  
Prof. Florian Dörfler, Automatic Control Laboratory, ETH Zürich

March 2021



# Acknowledgment

I would like to sincerely thank my supervisor at CU Boulder, Prof. Emiliano Dall'Anese, for his willingness to offer me such an interesting project, for his efforts to make me feel welcome despite the circumstances, for facilitating my stay in Boulder through the CU Europe-Colorado Mobility Program and for his unwavering support throughout the process - both in academic matters concerning my thesis and beyond. I would also like to thank Ana Ospina for the time and attention she committed to help me to solve issues and to review my thesis and presentation. I'd like to thank Prof. Florian Dörfler for facilitating me to conduct my Master Thesis overseas and for his advising. I am grateful also to Marina Gonzalez and Michael Koller at EKZ for allowing me to use company battery data. Finally, I'd like to thank my aunt, Lolo, for the encouragement, for the tea conversations, for the five-star vegan food and for being a dependable, supportive and close family member and friend to reach out to during my stay in the US.



# Abstract

The increased availability of sensing and computational capabilities in modern cyber-physical systems and networked systems has led to a growing interest in learning and data-driven control techniques. Learning-Based Model Predictive Control (LBMPC), i.e. the integration of learning methods in Model Predictive Control schemes, is one technique with potential applications for the control of dynamical systems under uncertain and stochastic conditions including humans in the control loop. In real-time applications of LBMPC, control solutions must be achieved in limited time, given the computational burden of both the function-learning and control mechanisms. Furthermore, models and functions associated with users must be learned on the fly from possibly parsimonious feedback. In this work, a framework is proposed for a LBMPC scheme in which little or no prior information is known regarding the dynamic state functions of multiple devices and the cost functions modeling satisfaction, comfort or sense of safety of users interacting with these devices. Gaussian Processes are advocated as a non-parametric nonlinear function modeling technique to enable a low sampling rate for black-box dynamic state and dissatisfaction functions. A regularized primal-dual gradient-based optimization algorithm is adapted to the discrete-time case and integrated with the LBMPC framework to facilitate convergence in a real-time setting with a Q-linear tracking error.

Considering the rapidly-evolving demands on the power grid as a result of increasing distributed energy resources, the proposed methodology is implemented for a typical Demand Response application, in which the power setpoints of a network of black-box thermostatically-controlled loads in a building are optimized to provide ancillary services to the grid under fluctuating power demands while minimizing costs incurred to the black-box users inhabiting the building. The methodology is also applied to the classic inverted pendulum problem for the purposes of comparing results across true and GP-learned dynamic state and cost functions.



# Contents

<b>Abstract</b>	iii
<b>Nomenclature</b>	ix
<b>1 Introduction &amp; Literature Review</b>	<b>1</b>
1.1 Problem Statement & Applications	1
1.1.1 Real-Time MPC with Unknown Cost & Dynamic State Functions	1
1.1.2 Applications to the Optimization of Energy Systems	3
1.2 Literature Review	3
1.2.1 Networked & Multiuser Dynamical Systems	3
1.2.2 Real-Time Learning of Black-Box Functions	5
1.2.3 Real-Time Model Predictive Control	6
1.2.4 Learning-Based Control Schemes	6
1.2.5 Online Primal-Dual Gradient Descent Optimization Methods	7
1.2.6 Modeling & Control of Thermostatically-Controlled Loads	8
1.3 Challenges	8
1.4 Contributions	8
1.5 Thesis Structure	9
<b>2 Gaussian Processes</b>	<b>11</b>
2.1 Bayesian Optimization	11
2.2 Gaussian Distributions	12
2.3 Gaussian Processes	13
2.4 Covariance Function	14
2.5 Hyperparameter Selection	15
2.6 Bayes' Theorem & Gaussian Processes	16
2.7 Predicting Outputs with Noise-Free Observations	17
2.8 Predicting Outputs with Noisy Observations	17
2.9 Measuring the Performance of a GP Prediction	18
2.10 Computing the First-Order Derivative of the Posterior Mean	18
2.10.1 Finite-Difference Method	18
2.10.2 Closed-Form First-Order Derivative of the Posterior Mean	19
2.11 Reproducing Kernel Hilbert Spaces	19
2.12 Gaussian Processes for Dynamic State Functions	20
2.13 Gaussian Processes for Cost Functions	21
2.14 Integrating GP Models with the MPC Scheme	22
<b>3 Model Predictive Control with Gaussian Process Regression</b>	<b>25</b>
3.1 Discrete-Time Model Predictive Control	25
3.2 Regularized Lagrangian Primal-Dual Gradient Optimization	29
3.2.1 Strong Convexity	29
3.2.2 The Gradient Method	30
3.2.3 The Projected Gradient Method	30



3.2.4	The Dual Problem	31
3.2.5	Regularized Lagrangian Function	34
3.2.6	Primal-Dual Gradient Method	34
3.3	Online Regularized Lagrangian Primal-Dual Gradient Optimization	36
3.3.1	Tracking Error	36
3.3.2	Warm-Starting	40
3.4	Sampling Timeline	40
<b>4</b>	<b>Application to Real-Time Optimization of Energy Systems</b>	<b>43</b>
4.1	System Model	43
4.1.1	Thermostatically-Controlled Load	44
4.1.2	Network	46
4.2	MPC Optimal Control Problem	48
4.3	Results of GP Predictions	51
4.3.1	TCL State Variation	52
4.3.2	TCL Stage Cost	53
4.4	Results of MPC Simulations	54
<b>5</b>	<b>Conclusions</b>	<b>65</b>
5.1	Gaussian Process-Learned Functions	65
5.2	MPC Schemes with GP-Learned Functions	66
5.3	Recommendations for Further Work	67
<b>A</b>	<b>Additional Application to an Inverted Pendulum System</b>	<b>69</b>
A.1	System Model	69
A.2	Results of GP Predictions	70
A.2.1	State Variation GP Predictions	70
A.2.2	Cost GP Prediction	71
A.3	Results of MPC Simulations	72
<b>B</b>	<b>Mathematical Background</b>	<b>79</b>
B.1	Probability & Probability Density	79
B.2	Joint, Marginal & Conditional Probability	79
B.3	Bayes' Theorem	80
B.4	Gaussian Identities	80
B.5	Convex Sets	81
B.6	Compact Sets	81
B.7	Mean Square Continuity & Differentiability	81
<b>C</b>	<b>Optimization Theory</b>	<b>83</b>
C.1	Convex Functions	83
C.2	Convex Optimization Problems	83
C.3	Karush-Kuhn Tucker (KKT) Optimality Conditions	83
C.4	Order of Convergence & Rate of Convergence	84
C.5	Regret	85
<b>D</b>	<b>System &amp; Control Theory</b>	<b>87</b>
D.1	Controllability, Observability & Stabilizability of a System	87
D.2	Discretizing a Nonlinear Continuous Dynamical System	87
D.2.1	First-Order Taylor Series Discretization	88
D.2.2	Runge-Kutte Discretization	88
D.3	Jensen's Inequality	88
D.4	Game Theory	88
D.4.1	Stackelberg Game	88
D.4.2	Social Welfare Problem	88

<u>D.4.3 Multi-Armed Bandit Problem</u> . . . . .	88
<u>D.5 Tikhonov Regularization</u> . . . . .	89



# Nomenclature

## Symbols

### Chapter 1

$\mathbf{u}_0^*$	Optimal Control Input for $k = 0$	[–]
$\mathbf{w}$	Exogenous Disturbance/Process Noise	[–]
$\mathbf{x}_0$	Initial System State	[–]
$\mathbf{n}$	Measurement Noise	[–]
$\hat{\mathbf{y}}$	Noisy Measurement	[–]
$\tilde{\mathbf{x}}_1$	Estimated System State	[–]

### Chapter 2

$M$	Prior Model of Unknown Function	[–]
$E$	Observed Evidence on Unknown Function	[–]
$N_{tr,0} \in \mathbb{N}$	Initial Number of Training Samples	[–]
$N_{tr} \in \mathbb{N}$	Number of Training Samples	[–]
$N_* \in \mathbb{N}$	Number of Test Samples	[–]
$\Delta t_l^d \in \mathbb{R}_{++}$	Stage Cost GP Sample Time for user $d$	[s]
$\Delta t_x^d \in \mathbb{R}_{++}$	State Variation GP Sample Time for device $d$	[s]
$n_y \in \mathbb{N}$	Number of Outputs	[–]
$n_x \in \mathbb{N}$	Number of States	[–]
$n_u \in \mathbb{N}$	Number of Inputs	[–]
$n_w \in \mathbb{N}$	Number of Disturbances	[–]
$X \in \mathbb{R}^{N_{tr} \times n_x}$	Training Inputs	[–]
$X_* \in \mathbb{R}^{N_* \times n_x}$	Test Inputs	[–]
$K$	Covariance Matrix	[–]
$\mathbf{f}$ or $f(\mathbf{x})$	True/Latent Function Values	[–]
$\mathbf{f}_*$ or $f(\mathbf{X}_*) \in \mathbb{R}^{N_*}$	Test Function Values	[–]
$\mathbf{y} \in \mathbb{R}^{N_{tr}}$	Output/Target values	[–]
$\boldsymbol{\mu}$	Mean Vector	[–]
$\mathcal{N}$	Gaussian/Normal Probability Distribution	[–]
$\sigma^2 \in \mathbb{R}_{++}$	Output Variance	[–]
$\mathbf{l} \in \mathbb{R}_{++}^{n_x}$	Characteristic Length Scale	[–]
$\sigma_n^2 \in \mathbb{R}_+$	Measurement Noise	[–]
$\boldsymbol{\theta} \in \mathbb{R}^{1+n_x}$	Hyperparameter Vector	[–]
RSS $\in \mathbb{R}$	Residual Sum of Squares	[–]
TSS $\in \mathbb{R}$	Total Sum of Squares	[–]

$\text{Score}_{\text{GP}} \in \mathbb{R}$	GP Score	[-]
$\delta \in \mathbb{R}_{++}$	Infinitesimal Difference	[-]
$\mathcal{H}$	Hilbert Space	[-]
$\mathcal{H}_k$	Reproducing Kernel Hilbert Space of Kernel $k$	[-]
$f_i^d$	$i^{\text{th}}$ Dynamic State Function of Device $d$	[-]
$h_i^d$	Prior Component of $i^{\text{th}}$ Dynamic State Function of Device $d$	[-]
$m_i^d$	Prior Component of Cost Function of User $d$	[-]
$g_i^d$	Learned Component of $i^{\text{th}}$ Dynamic State Function of Device $d$	[-]
$j_i^d$	Learned Component of Cost Function of User $d$	[-]
$n_x^d$	Number of States of Device $d$	[-]
$n_u^d$	Number of Control Inputs of Device $d$	[-]
$n_w^d$	Number of Exogenous Disturbances of Device $d$	[-]
$\mathbf{x}_k^d \in \mathbb{R}^{n_x^d}$	State Vector of Device $d$ at Time-Step $k$	[-]
$\tilde{x}_{i,k}^d \in \mathbb{R}^{n_x^d}$	$i^{\text{th}}$ Measured State of Device $d$ at Time-Step $k$	[-]
$\tilde{l}_k^d \in \mathbb{R}$	Measured User Dissatisfaction of Device $d$ at Time-Step $k$	[-]
$\kappa$	Measurement Gaussian Noise	[-]

### Chapter 3

$k \in \mathbb{N}$	Sample Number	[-]
$t \in \mathbb{N}$	Continuous Time Variable	[-]
$N \in \mathbb{N}$	Length of MPC Horizon	[-]
$T \in \mathbb{N}$	Number of Simulation Time-Steps	[-]
$\Delta t \in \mathbb{R}_{++}$	Discrete MPC Sample Time	[s]
$\mathbf{u}_k^d \in \mathbb{R}^{n_u^d}$	State Vector of Device $d$ at Time-Step $k$	[-]
$\mathbf{w}_k^d \in \mathbb{R}^{n_w^d}$	State Vector of Device $d$ at Time-Step $k$	[-]
$\mathbf{z}_k^d \in \mathbb{R}^{n_x^d + n_u^d + n_w^d}$	State, Input and Disturbance Vector of Device $d$ at Time-Step $k$	[-]
$Z^d \in \mathbb{R}^{N_{tr} \times (n_x^d + n_u^d + n_w^d)}$	State Vector, Inputs and Disturbances of Device $d$ at Time-Step $k$	[-]
$\mathcal{X}_k \subseteq \mathbb{R}^{n_x^d}$	Feasible Set of States at Stage $k$	[-]
$\mathcal{X}_f \subseteq \mathbb{R}^{n_x^d}$	Terminal Feasible Set of States	[-]
$\mathcal{U}_k \subseteq \mathbb{R}^{n_u^d}$	Feasible Set of Control Inputs at Stage $k$	[-]
$n_g^0 \in \mathbb{N}$	Number of Explicit Inequality Constraints of the System at each Stage	[-]
$n_g^d \in \mathbb{N}$	Number of Explicit Inequality Constraints of Device $d$ at each Stage	[-]
$n_g := \sum_{d=1}^D n_g^d \in \mathbb{N}$	Number of Explicit Inequality Constraints of all Devices at each Stage	[-]
$\mathbf{g}^0 : \mathbb{R}^{N n_y} \mapsto \mathbb{R}^{N n_g^0}$	Explicit Inequality Constraints of the System	[-]
$\mathbf{g}^d : \mathbb{R}^{N n_x^d} \mapsto \mathbb{R}^{N n_g^d}$	Explicit Inequality Constraints of Device $d$	[-]
$\mathbf{G} : \mathbb{R}^{N(n_x + n_y)} \mapsto \mathbb{R}^{N(n_g^0 + n_g)}$	All Explicit Inequality Constraints	[-]
$\mathbf{F} : \mathbb{R}^{N(n_x + n_u)} \mapsto \mathbb{R}^{N n_x}$	All Dynamic State Equality Constraints	[-]
$\mathbb{E}$	Euclidean Set	[-]
$\alpha_\tau$	Gradient Method Primal Stepsize	[-]
$\eta_\tau$	Gradient Method Inequality Dual Stepsize	[-]
$\epsilon_\tau$	Gradient Method Equality Dual Stepsize	[-]
$\nu$	Gradient Method Tolerance Parameter	[-]
$\nabla$	Gradient Operator	[-]
$\nabla^2$	Hessian Operator	[-]

$J_f$	Jacobian Matrix for Function $f$	[–]
$N_C$	Normal Cone	[–]
$\mathcal{P}$	Projection Operator	[–]
$I$ or $I_n$	Identity Matrix	[–]
$B_n(r)$	$n$ -Dimensional Ball of Radius $r$	[–]
$\mathcal{L} : \mathbb{R}^{N(n_x+n_u+n_g^0+n_g+n_x)} \mapsto \mathbb{R}$	Lagrangian Function	[–]
$\mathcal{L}^r : \mathbb{R}^{N(n_x+n_u+n_g^0+n_g+n_x)} \mapsto \mathbb{R}$	Regularized Lagrangian Function	[–]
$\mathbf{z} \in \mathbb{R}^{N(n_x+n_u)}$	Primal Variables	[–]
$\boldsymbol{\lambda} \in \mathbb{R}^{N(n_g^0+n_g)}$	Inequality Dual Variables	[–]
$\boldsymbol{\lambda}_{prior} \in \mathbb{R}^{N(n_g^0+n_g)}$	Inequality Dual Variable Priors	[–]
$\boldsymbol{\mu} \in \mathbb{R}^{Nn_x}$	Equality Dual Variables	[–]
$\boldsymbol{\mu}_{prior} \in \mathbb{R}^{Nn_x}$	Equality Dual Variable Priors	[–]
$\mathcal{A}$	Real-Time/Online Algorithm	[–]
$\mathbb{L}$	Bounded Family of Cost Functions	[–]
$r_\tau : \mathbb{R}^{N(n_x+n_u)} \mapsto \mathbb{R}$	Instantaneous Regret	[–]
$R_{\text{MaxIter}} : \mathbb{R}^{N(n_x+n_u)} \mapsto \mathbb{R}$	Cumulative Regret	[–]

## Chapter 4

$D$	Number of Controllable Devices	[–]
$W$	Number of Uncontrollable Devices	[–]
$T_k$	Zone Temperature at Time-Step $k$	[°C]
$\underline{T}_k, \bar{T}_k$	Minimum and Maximum Zone Temperature at Time-Step $k$	[°C]
$P_k$	Power Consumed at Time-Step $k$	[kW]
$\bar{P}_k$	Maximum Power Consumed at Time-Step $k$	[kW]
$P_{fan,k}$	Power Consumed by Fan at Time-Step $k$	[kW]
$P_{ch,k}$	Power Consumed by Chiller at Time-Step $k$	[kW]
$\dot{m}_k$	Air Flow Mass at Time-Step $k$	[kW]
$T_{da,k}$	Discharge Air Temperature at Time-Step $k$	[kW]
$T_{o,k}$	Outdoor Air Temperature at Time-Step $k$	[kW]
$Q_{s,k}$	Solar Irradiance at Time-Step $k$	[kW]
$Q_{i,k}$	Internal Irradiance at Time-Step $k$	[kW]
$y_{ref,k}$	Reference Power at Time-Step $k$	[kW]
$\beta$	Power Deviation Cost Coefficient	[–]

## Appendix A

$\theta_k$	Angle relative to Vertical Position at Time-Step $k$	[rad]
$\dot{\theta}_k$	Rate of Change of $\theta$ at Time-Step $k$	[rad/s]
$\tau_k^\alpha$	Controlled Torque acted on Pendulum at Time-Step $k$	[Nm]
$\tau_{d,k}^\alpha$	Exogenous Torque imposed on Pendulum at Time-Step $k$	[Nm]
$m$	Mass of Pendulum	[kg]
$l$	Length of Pendulum	[m]
$\eta_f$	Friction Parameter	[Nm/s/rad]
$g$	Gravitational Constant	[m/s <sup>2</sup> ]

## Indices

$d$	Device
$k$	Time-Step in MPC Horizon
$k_0$	Absolute Time-Step in MPC Simulation
$t$	Training/Test Sample Index
$r$	Regularized (Lagrangian Function)
$f$	Terminal Set/Cost
$0$	System Cost/Constraint
$i$	State Index
$tr$	Training Data
$*$	Test Data
$\star$	Optimal Point

## Acronyms and Abbreviations

OCP	Optimal Control Problem
MPC	Model Predictive Control
DTMPC	Discrete-Time Model Predictive Control
NMPC	Nonlinear Model Predictive Control
LBMPC	Learning-Based Model Predictive Control
RMPC	Robust Model Predictive Control
SMPC	Stochastic Model Predictive Control
LQMPC	Linear-Quadratic Model Predictive Control
BO	Bayesian Optimization
GP	Gaussian Process
ARD	Automatic Relevance Determination
RKHS	Reproducing Kernel Hilbert Spaces
CT	Continuous-Time
DT	Discrete-Time
RBF	Radial Basis Function
SE	Squared-Exponential
TCL	Thermostatically-Controlled Load
ESS	Energy-Storage System
DER	Distributed Energy Resource
DR	Demand Response
QP	Quadratic Program
mpQP	Multi-Parametric Quadratic Program
LQ	Linear Quadratic
TDO	Time-Distributed Optimization
IoT	Internet-of-Things
KKT	Karush-Kuhn-Tucker
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
OED	Optimal Experiment Design

RHGC Receding Horizon Gradient-based Control





# Chapter 1

## Introduction & Literature Review

### 1.1 Problem Statement & Applications

#### 1.1.1 Real-Time MPC with Unknown Cost & Dynamic State Functions

Model Predictive Control (MPC) is a powerful framework for controlling a dynamic system to meet performance and safety objectives by predicting responses of the system to different inputs, and optimizing over these inputs for some future time horizon. As systems become more complex however, the time required to solve this problem may supersede the time available before a control action must be implemented. In this case, we must settle for sub-optimal solutions attained before the optimization algorithm has converged with real-time control schemes.

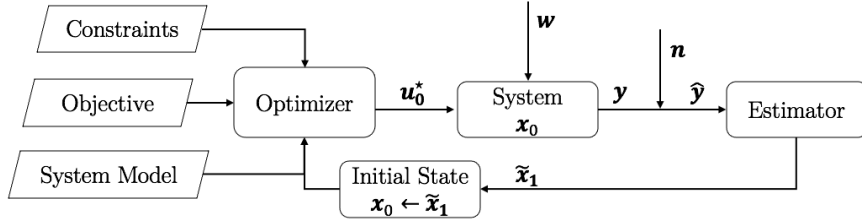
In theory, we can control any controllable (see Def. [D.1.1](#)) system given that we have a well-defined objective and a precisely identified system. However, real-life systems and objectives are dynamic and subject to significant uncertainty. The assumed models used in the control scheme to describe the system and the objective may deviate considerably from the true models as a result of environmental disturbances, restrictive model classes or insufficient modeling data. The hidden true functions may express complex relationships which are sparse, nondifferentiable, or even unavailable in closed-form expressions, such that they are ill-suited to gradient-based numerical optimization methods [\[1\]](#).

There are cases, namely when a human is involved in the control loop, where the cost function must be specific and adaptive to the user in question. In these contexts, the modeled cost function, which could be derived from dated or inaccurate assumptions or generalized from tests on restrictive samples of users to the greater population, may not correspond closely to the true cost function of the real user. There are cases when complex systems cannot be well described by available synthetic models or approximations, and furthermore when these systems evolve over time due to degradation or changes in unobservable states. There are cases, where users or systems cannot be “taken offline” to conduct the extensive testing required to identify the true objective or a high-fidelity system model. Even if tests are possible, they are likely to be expensive in terms of time and effort. Nonlinear, nonparametric identification methods offer a potential solution to these challenges, as they are well-suited to identifying bias-free objective and system models in an online fashion [\[2\]](#).

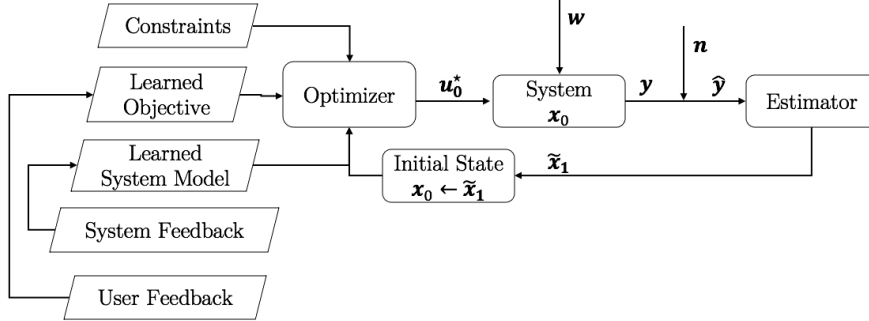
When allowing for unknown or uncertain functions in a control scheme, a trade-off to address is the requirement to express the objectives of the controller and the dynamics of the system in a way that closely reflects true behavior, while exhibiting certain properties that facilitate solving the problem. The frameworks of robust MPC (RMPC) [\[3\]](#) and stochastic MPC (SMPC) [\[4\]](#) explicitly consider different sources of uncertainty in their formulation, ensuring constraint satisfaction for certain classes of disturbances or uncertainties. One of the limitations of these control methodologies however, is their strict division between a *design phase* and an *application phase*. The former is generally executed offline by a control engineer, and the latter describes the subsequent execution of the controller, during which its formulation remains largely static [\[1\]](#).

Under these paradigms, there is limited scope for *adaptive* control of the system, whereby the models describing the problem may be updated online in response to new information.

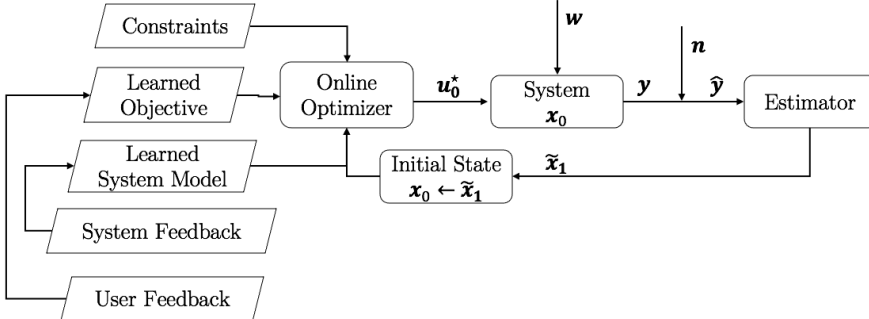
The work of this paper diverges from nominal MPC schemes (illustrated in Fig. 1.1a) in two ways. Firstly, while there exists extensive literature on nominal MPC methods in which the deterministic models for the cost and system dynamics are known, we consider the case in which a) the system affects the user, but no model for the perspective of the user exists and b) it is costly or infeasible to formulate a physics-based model of the system with thorough offline testing. Therefore nonparametric learning methods are advocated in this work to model the cost to the user and the response of the system for different possible control actions, system states and environmental disturbances (illustrated in Fig. 1.1b). Secondly, while many existing MPC schemes assume optimization to convergence (*batch* optimization), we advocate a departure from the traditional setting to address time-complexity constraints, a limited time-frame in which a solution is required or unobservability of exogenous variables by implementing a real-time MPC scheme (illustrated in Fig. 1.1c) with an *online* optimization algorithm.



(a) Nominal Batch MPC



(b) Learning-Based Batch MPC



(c) Real-Time Learning-Based MPC

Figure 1.1: MPC Schemes

### 1.1.2 Applications to the Optimization of Energy Systems

Global energy supply is undergoing a rapid transition from conventional fossil fuel sources to renewable alternatives. This places new demands on the existing electricity grid, which must continue to balance unpredictable energy demand with stochastic renewable energy supply. A failure to adapt to this changing energy landscape can result in extreme price volatility and additional strain on the existing grid infrastructure. Additionally, the electricity grid must accommodate an increasing number of distributed energy systems. While some of these devices pose a challenge, others, such as *controllable loads* offer an opportunity for intelligent adaptation of the grid using principles of control and learning theory. Demand response (DR) strategies, for example, have the potential to provide ancillary services to the grid by enabling controllable energy devices to support power and voltage balance services at different time-scales [5, 6, 7, 8, 9]. DR schemes can be formulated as *multiuser optimization problems*, in which the objective is to maximize the contribution of a network of devices (or *users*) to the grid based on real-time requirements, while minimizing the costs incurred (or alternatively, maximizing the utility provided) to the users of these devices within constraints that can couple multiple devices [10, 11]. Clearly, such approaches necessitate integrating user-preferences in the control loop, which cannot always be modeled by closed-form expressions, generalized over a population of users or assumed to be invariant with respect to time and environmental variables.

The MPC framework has been established in process control, automotive systems, robotics and chemical processes. In this work, the algorithmic framework presented is validated by controlling a network of distributed energy systems in building, where it is not possible to obtain a synthetic model of the cost incurred to the users of each device. We therefore implement Gaussian Processes to learn the dissatisfaction of users from real-time user feedback. In addition, it is not feasible to regularly close and empty the building to perform extensive testing for the purposes of maintaining an up-to-date dynamic state model of each device. Instead, we propose leveraging feedback from the individual devices to generate a time-varying Gaussian Process model for each one which reflects changes in the device and its environment over time.

The MPC scheme is a natural choice for energy systems as it meets the real-time requirement of the control scheme, allows for the integration of constraints on each device as well as on their coupling and allows for a composite cost (or *objective*) function which includes both terms specific to individual users as well as for their coupling. The receding horizon nature of MPC provides a framework to ensure recursive feasibility and stability of the control scheme. A real-time MPC scheme such as that employed by a distributed network of energy systems requires a highly-efficient optimization algorithm which can achieve satisfactory results within very few (ideally one) iterations. In this work, a primal-dual gradient-based optimization method is employed for this purpose due to the Q-linear convergence properties achieved by the tracking error of its online implementation.

## 1.2 Literature Review

In this section, we will outline the key findings in a representative collection of papers applying control and/or GP learning strategies to multiuser problems with unknown systems and/or cost functions.

### 1.2.1 Networked & Multiuser Dynamical Systems

The multiuser problem describes a constrained optimization problem specified by a set of users, an objective function given by a sum of user-specific utility functions and a collection of linear constraints that couple the users' decisions. The authors of [12] consider the problem in which the user decisions are coupled both in the objective function by a convex coupling cost and by convex nonlinear constraints which are not necessarily separable by user. An approximation of the multiuser problem is obtained with a Tikhonov regularization (see Appendix D.5) and a gradient-based distributed optimization algorithm is used to solve this approximation. Different step lengths

are used across users as well as across the primal-dual space. An estimate of the error achieved between the true optimal function values and those resulting from the regularized variant is derived. In this distributed setting, it is assumed that each user a) knows only its own objective function and feasible sets and b) can modify only its own decision variables but can observe the decisions made by other users. The approach of this *limited coordination* algorithm is for each user to update its own decision variables. The multiuser optimization structure is adopted in this work for the case where a network of devices must be controlled in real-time, and the device dynamics and user cost functions are unknown.

The problem of DR control systems which consider user feedback is addressed in [13]. A known time-varying engineering cost is combined with an unknown discomfort function for each device in the system to formulate a composite cost function which balances system-level operational objectives and user dissatisfaction objectives. Measurements of electrical quantities are used to estimate the gradient of the known engineering cost, such that measurements of the disturbances which influence the electrical quantities are not required. A shape-constrained Gaussian Process is used to learn the unknown user dissatisfaction function, resulting in a strongly convex and smooth approximation that can handle asynchronous and noisy data. A projected gradient method is used to solve the single-stage problem at each time-step. In this thesis, a similar problem is addressed using the MPC framework with integrated black-box dynamic state models.

The real-time coordination of networked distributed energy resources (DERs) is studied in [14], where each DER device (which is connected to a node in the network) may have continuous and discrete power setpoints broadcasted to it by a network operator and constraints on distribution lines couple the power injections and state dynamics of individual devices. Real-time implementation is facilitated by assuming a linear relationship between the electrical quantities of interest which couple the nodes and the power flow injections. A framework is provided to integrate both devices with convex and compact power setpoints (or control inputs) and discrete power setpoints. The coupling variables are constrained by an affine network-level inequality. The cost function for each node is defined as the sum of the costs associated with the individual devices connected to that node and is assumed to be strongly convex. Both batch and online regularized primal-dual gradient-descent algorithms are presented with unique saddle points and linear convergence properties. A distributed stochastic implementation is proposed in the form of a Stackelberg game (Appendix D.4.1), whereby the decomposability of the Lagrangian function is leveraged such that each node can update the primal variable power setpoints of the devices connected to it, and the network operator can in turn update the dual variables. The algorithm is validated using synthetic models of HVAC and energy-storage systems. In this work, we employ a similar composite objective function including GP-learned (but not restricted to shape-constrained) user-specific utility functions in a real-time MPC scheme.

In [9], a distributed optimization algorithm is proposed for solving optimal control problems in multi-building networks which are coordinated by a DR program. The problem is modeled as a distributed convex optimization problem with separable cost functions and coupled affine equality constraints. A variant of the Augmented Lagrangian based Alternating Direction Inexact Newton (ALADIN) optimization method is employed and a convergence guarantee is provided. The state dynamics of each building are modeled by discrete linear equations, where the states represent the temperatures of the thermal zones and the inputs represent the thermal cooling energy input. The objective for each individual building is to track a given target room temperature while minimizing the energy input required to do so. A strongly convex but non-smooth quadratic programming (QP) problem is formulated which aims to minimize the economic cost of the electricity, track the reference temperatures in each zone and minimize the energy input required to achieve this. In the proposed algorithm, a decoupled QP is solved locally at each building and the solution is communicated to the grid operator. The grid operator then solves an equality-constrained QP and communicates the relevant solution to each building. The local optimal control problem (OCP) is equivalent to a multi-parametric QP (mpQP) and so the solution maps are piecewise affine. The operator OCP is a QP without inequality constraints, and thus has an analytical solution. MPC is chosen as a controller scheme which meets the real-time requirements of the problem and gradient-based primal-dual decomposition optimization methods

are leveraged to develop an efficient online solver. Additionally, warm-starting is employed to facilitate online tracking of the true MPC optimal trajectory. The proposed online MPC strategy succeeds in balancing the voltage surge of the building network while tracking a reference temperature for each zone at a low power input. In this work, a similar case study is employed, with unknown functions describing the dynamics of each building and users involved in the control loop.

## 1.2.2 Real-Time Learning of Black-Box Functions

In a *multiarmed bandit problem* (see Appendix [D.4.3](#)), the controller attempts to optimize a cost function that it is simultaneously learning. In [\[15\]](#), the GP-UCB algorithm is proposed to solve this problem, where inputs are selected which achieve a high upper confidence bound on functions sampled from either a Gaussian Process or a reproducing kernel Hilbert space (RKHS). Both the potential information gain (exploration) and the potential to find the extrema of the function (exploitation) are considered by selecting function inputs which maximize the posterior mean and the posterior standard deviation of the unknown function, with relative weighting expressing the trade-off between the dual objectives. The authors succeed in establishing a novel link between optimization of a GP-sampled function and optimal experiment design (OED) by deriving bounds on the cumulative regret (see Appendix [C.9](#)) of the algorithm and proving sublinear regret bounds for many commonly used kernel functions.

When implementing Gaussian Processes and the associated covariance kernel functions in practice, one cannot assume that they know the kernel hyperparameters in advance and misspecification can result in convergence to poor local optima. The authors of [\[16\]](#) address this problem by proposing an algorithm for Bayesian optimization with unknown hyperparameters and deriving bounds on the error achieved. The so-called Adaptive GP-UCB algorithm is proposed for the case where neither the norm bound nor the length-scales of a stationary covariance kernel are known. The function space associated with the hyperparameters is slowly expanded over time, resulting in a BO algorithm that is provably no-regret when the covariance kernel hyperparameters are unknown.

The majority of the computational burden of making predictions with GP-modeled functions results from the inversion of the training data covariance matrix ( $\mathcal{O}(N_{tr}^3)$ ). This can be a barrier to learning functions in a real-time control scheme. The authors of [\[17\]](#) developed a) a ‘sliding-window’ approach to fix the dimensions of the covariance matrix while updating it with new training samples online and more critically, b) an efficient means of computing the matrix inverse requiring only the previous inverted matrix and the new training samples, resulting in a time-complexity of  $\mathcal{O}(N_{tr}^2)$ . Other methods for more efficient covariance matrix inversion include a ‘sparsification’ procedure proposed in [\[18\]](#), in which a candidate training sample is only admitted to the training set if its image in feature space cannot be sufficiently approximated by combining the existing training samples; or the application of principal component analysis (PCA) proposed in [\[19\]](#), whereby the dominant eigenvectors of the covariance matrix are extracted using singular value decomposition (SVD).

An alternative approach to learning unknown nonlinear functions given input and output samples is presented in [\[20\]](#). A continuous-time recurrent neural net with a hyperbolic tangent activation function (i.e. a nonlinear function which determines whether a neuron should be ‘activated’ or not before sending its input to the next layer of neurons or finalizing it as an output) is developed which approximately reproduces the unknown function input-output behavior. The temporal-nature of the unknown dynamic state model considered is leveraged in the *recurrent* neural network approach. This requires connections between nodes which form a directed graph along a temporal sequence (i.e. previous states, inputs and outputs to subsequent ones). The key element of recurrent neural networks is that they have memory - they can take information from prior neuron inputs to influence the input and output of the current layer of neurons. The model class considered is a state-space linear model with unknown parameters filtered by the activation function. A number of neurons equal to the number of states is employed in a learning procedure that generates these parameters within a bounded set based on pairs of inputs and outputs observed. A bound on the error of the approximated model is derived, which depends on

a minimum number of samples, a sufficiently high number of available derivatives of the output, the time period over which the model is defined and the bound on the model parameters. While this approach requires only input and output values of the black-box function under study, the drawback is the high minimum number of samples required to guarantee a bound on the modeling error. Additionally, for the purposes of user-specific utility functions, we cannot generalize a model generated at a high computational expense for a small sample of users to the general population.

### 1.2.3 Real-Time Model Predictive Control

*Time-Distributed Optimization* (TDO) [21] or *Real-Time Iteration* (RTI) [22] is an approach to reducing the computational burden of a MPC scheme. With this method, optimization iterations are distributed over time by maintaining a running solution estimate and updating it at each sampling instant. The central concept of this method is that an iterative optimization method can be truncated to produce a sub-optimal MPC feedback law while maintaining stability and constraint satisfaction and reducing the computational burden of the controller. The authors of [21] perform a detailed systems theoretic analysis of this control scheme to identify and analyze multiple mechanisms for ensuring stability of the closed-loop system. They consider an input-constrained Linear-Quadratic MPC (LQMPC) using primal gradient-based optimization methods and derive analytic expressions for the MPC and optimizer gains, a sufficient condition for asymptotic stability and a corresponding iteration bound. Strategies identified include increasing the number of solver iterations, preconditioning the OCP, tuning the cost function and reducing the prediction horizon. The authors of [22] provide a proof for the asymptotic stability for a class of real-time optimization methods for nonlinear MPC (NMPC) by constructing a Lyapunov function for the system-optimizer dynamics for the case where a single iteration is executed to solve the problem at each time-step.

### 1.2.4 Learning-Based Control Schemes

The authors of [1] review solutions to the problem of learning-based control for three categories: a) automatic improvement of the dynamic state model using data measurements, b) learned parameterization of the MPC problem and c) leveraging MPC to augment learning-based control schemes with constraint satisfaction properties. A general OCP for uncertain costs and system dynamics is formulated, in which a) two sources of *uncertainty* or *error* are considered: time-invariant parametric uncertainty and time-variant uncertainty derived from disturbance or process noise; b) the deterministic objective function is replaced by the *expected* value of the sum of stage cost functions over all uncertainties and c) the deterministic constraints on the states and control inputs are replaced by joint (for the feasible state space and the feasible control input space) *chance constraints* whereby the constraints must be satisfied with a given minimum probability. This results in an equivalent *stochastic* optimal control problem. Given the challenges involved in directly solving such a problem MPC is proposed as a tractable approximate solution strategy. An extensive review of existing learning-based MPC paradigms designed to solve this problem are reviewed.

A safe learning-based MPC scheme is presented in [23]. The purpose of the scheme is to efficiently explore the input space of the model within the context of safety-critical applications. *Safety* in this context is defined in terms of recursive feasibility and robust constraint satisfaction. This is achieved by guaranteeing the existence of feasible return trajectories to a safe region of the state-space with high probability. It is assumed that: a) the unknown function belongs to a RKHS model class, b) the unknown error function has a bounded norm (as measured by the norm induced by the continuously differentiable and unique kernel of the RKHS), c) the system is subject to polytopic state and control constraints and d) we have access to a backup controller that guarantees that we remain inside a given safe subset of the state space once we enter it. An inverted pendulum system is considered to validate the proposed MPC strategy. A *prior* model of the system dynamics is taken to be a linearized synthetic approximation of the true function with inaccurate parameterization and the model error is then learned with Gaussian Processes, where the errors of these GP approximations are contained by confidence-ellipsoids. The result is a learning-based MPC scheme that can provide provably high-probability safety guarantees. The

work of [23] is extended in this thesis to include black-box cost functions, *time-varying* dynamic state functions and time-varying constraints within a real-time MPC context.

Without employing additional matrix inversion techniques, the complexity of training and predicting with Gaussian Processes is  $\mathcal{O}(N_{tr}^3)$ , due to the necessary inversion of the training data covariance matrix. It is thus desirable to obtain the most accurate GP model possible with the least amount of training data. To this end, in [24] an optimal subset of the training set is selected. A sub-problem is solved to maximize the information gain over all of the auto-regressive outputs, control inputs and exogenous disturbance inputs available in the training set. A chance constraint guarantees a minimum probability that the output is within a given set. A zero-variance method is used which assumes the auto-regressive outputs to be equivalent to their expected values, and therefore this strategy does not propagate uncertainty through the multi-step simulation. The algorithms developed were applied to an energy management case-study during a DR event for large electricity consumers.

Gaussian Processes are used in conjunction with MPC to learn a nonlinear dynamical system in [25]. The proposed MPC scheme considers the variance (or uncertainty) of GP approximation in predictions of system behavior over the horizon. The resulting GP model describes both the expected system dynamics as well as the confidence in these predictions. Once this GP model is integrated into a MPC scheme, the closed-loop system is designed to avoid region with large variance (in favor of safety) at the expense of a greater cost function (at the expense of performance). The control scheme is demonstrated with a pH process control example.

In [26], a learning-based robust controller is proposed in which the initial (prior) model of the system is updated using Gaussian Processes to approximate the model error such that the uncertainty associated with the model gradually decreases over time. A stabilization problem is considered and the GP model is linearized around an operating point to obtain a robust linear controller. The mean, the variance and the closed-form expressions for the derivatives of the mean and variance with respect to the inputs are utilized to formulate a linear state-space model of the system as well as an uncertainty model.

### 1.2.5 Online Primal-Dual Gradient Descent Optimization Methods

The authors of [27] address the synthesis and analysis of regularized primal-dual gradient methods to track a Karush-Kuhn-Tucker (KKT) trajectory. Sufficient conditions for the online algorithm are derived under certain conditions, asymptotic bounds for the tracking error are derived and analytical convergence is derived for a continuous-time version of the algorithm. Additionally, sufficient conditions such that the KKT trajectories will not bifurcate or merge are proposed. This work is extended in [28] to include equality constraints. The proposed algorithm in [28] is adapted in this thesis to serve a discrete-time MPC control scheme, to use the closed-form expressions for the derivative of the posterior mean of a Gaussian Process in the iterate updates and to allow for more than a single iteration. Furthermore, it is verified with a case study on a system of energy systems.

In [29], an online primal-dual projected-gradient method is applied to solve a time-varying optimization problem associated with a networked system. The time-varying Lagrangian function of the problem is regularized and thus designed to be strongly convex (concave) in the primal (dual) variables. Hence, the optimizer is unique but does not coincide with the optimizer for the original Lagrangian function. Measurement feedback is employed, whereby measurement of the control inputs and outputs are used in place of the last issued control inputs or postulated model for the measurement. This technique lends itself to a distributed implementation and does not rely on measurements of the exogenous inputs which may affect the outputs. Proofs of average regret bounds and linear convergence are provided. The time-varying network model from [29] is adapted in this work to consider a system constrained by nonlinear dynamic state functions and controlled by a MPC scheme.

In [30], the authors deal with online convex optimization problems with time-varying black-box cost and constraint functions. The cost function is learned using feedback at given test points and the constraints are learned, and possibly simultaneously violated, upon making



decisions. Proposed applications for this work include online network tasks e.g. Internet-of-Things (IoT), where online decision-makers (or control schemes) must flexibly adapt to unpredictable user preferences and resource limitations. The bandit online saddle-point (BanSaP) control scheme proposed can adapt in response to feedback from multiple users and the environment. The authors measure the performance of this algorithm using dynamic regret and *fit* (accumulated constraint violation). The algorithm is designed to be a *gradient-free* lightweight approach which is amenable to low-power distributed devices. The goal of the algorithm is to find a sequence of solutions which minimizes the aggregate time-varying cost function and to ensure that the time-varying constraints are satisfied in the long run on average. The limitations of this approach are that the unknown functions are assumed to be convex and measurements of the disturbances, on which the constraint function is dependent, are tantamount to learning the function.

In [31], the Receding Horizon Gradient-based Control (RHGC) scheme is proposed to optimize a time-invariant linear dynamical system with an associated time-varying convex stage cost in an online fashion, in which accurate predictions of the stage cost in a finite lookahead window are available to use in gradient calculations. Upper bounds for the dynamic regret of the online algorithm based on the standard gradient-descent and triple-momentum accelerated gradient-descent methods are derived and the proposed algorithm is validated in a path-tracking experiment. The limitations of this method are that the regret properties derived assume a deterministic linear system and the availability of accurate predictions of the stage cost function over a future time-window.

### 1.2.6 Modeling & Control of Thermostatically-Controlled Loads

In [32], the application of MPC to an aggregation of thermostatically-controlled loads (TCLs) for the purposes of short-term DR ancillary services is demonstrated. The authors employ an aggregated state-space model which explicitly models TCL heterogeneity and show that the parameters required for the aforementioned aggregated model can be determined from the parameters of individual TCLs or by observation of their temperature dynamics. Similarly, in [33, 34, 35], energy systems involving TCLs have been studied using postulated models for the TCL devices. In this work, however, we use Gaussian Processes to learn these models in real-time.

## 1.3 Challenges

While real-time MPC is a powerful tool when the the system, constraints and objectives are well-defined, additional work is required in contexts in which these conditions do not hold. Static models of systems and cost functions do not capture unpredictable environments, changing user-preferences or evolving systems. Additionally, human-in-the-loop control schemes demand a more nuanced approach to modeling user preferences than is provided by typical synthetic functions. The challenges faced by real-time learning-based MPC schemes, which are addressed in this work, are as follows:

- Distributed control schemes may include devices for which the dynamics and associated user dissatisfaction are time-varying black-box functions, and so models based on stationary, synthetic models may not be adequate.
- The sampling required to learn device-specific and user-specific functions may be costly and/or require a low sampling rate.
- Real-time learning-based MPC requires a highly efficient optimization algorithm into which learned models can be efficiently integrated.

## 1.4 Contributions

In this work, we apply an online primal-dual gradient-based optimization algorithm to a real-time MPC scheme involving a network of distributed devices, in which the dynamics and cost functions

of the individual devices and associated users are unknown. The proposed real-time learning-based MPC methodology is used to solve a network of thermostatically-controlled loads (TCLs) in a building network by learning the TCL temperature dynamics and the discomfort of the users while controlling the cooling power to provide ancillary services to the power grid while satisfying the preferences of the building inhabitants. The contributions of this work can be summarized as follows:

- Gaussian Processes are examined and implemented as a means of integrating black-box function models into a real-time gradient-based MPC scheme.
- The real-time learning-based MPC framework is adapted to account for unknown, time-varying and possibly nonlinear dynamic state and user dissatisfaction functions.
- The proposed algorithm is implemented to optimize a network of energy systems and an inverted pendulum system.

## 1.5 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 outlines the background theory behind modeling of black-box functions using Gaussian Processes and formulates the necessary notation and methodology. Performance metrics for GP models are defined and methods for defining the covariance kernel hyperparameters, choosing the prior functions, collecting the training data and promoting safety for the control scheme adopting the GP models are described.
- Chapter 3 outlines the background theory behind discrete-time model predictive control (DTMPC), formulates the standard DTMPC problem and an adapted variant with integrated Gaussian Processes. Additionally, it outlines the theory behind the primal-dual gradient optimization method used and presents the real-time MPC with Gaussian Processes algorithm. Performance metrics for online control schemes and options for improving online tracking performance are described.
- Chapter 4 formulates the building control problem used to validate the proposed methodology and provides results and discussion on the GP models, batch MPC simulations and real-time MPC simulations.
- Chapter 5 assesses the results of the case study, highlights key benefits and drawbacks of the proposed and tested methodology and provides recommendations for further work.
- Appendix A describes a second experiment conducted on an inverted pendulum system with the proposed control methodology. Results and discussion are provided on the GP models and batch MPC simulations for both the true functions and the GP-modeled functions.
- Appendix B defines mathematical terms and concepts referenced to within the main text.
- Appendix C defines terms and concepts relevant to optimization theory referenced to within the main text.
- Appendix D defines terms, techniques and concepts associated with systems and control theory referenced to within the main text.



## Chapter 2

# Gaussian Processes

In this chapter, the theoretical knowledge fundamental to learning unknown functions with Gaussian Processes is outlined and the framework used in this work to learn dynamic state models and cost functions is given. Additional notes on relevant mathematical background are provided in [B](#).

### 2.1 Bayesian Optimization

Bayesian optimization (BO) is a method for finding the extrema of an unknown objective function that is expensive to evaluate (see [36](#) for a comprehensive tutorial). It is useful in scenarios in which the closed-form expression for the function is unknown, but noisy or noise-free observations of the true function can be obtained at sampled inputs in the domain.

*Bayes' Theorem* is employed to combine prior beliefs about the unknown function with new evidence to approximate the function by its *posterior*. It states that the *posterior* probability of a model  $M$  given *evidence*  $E$  is proportional to the *likelihood* of  $E$  given *model*  $M$  multiplied by the *prior* probability of the model  $M$  (see [2.1](#)).

$$\underbrace{P(M|E)}_{\text{posterior distribution}} \propto \underbrace{P(E|M)}_{\text{likelihood}} \underbrace{P(M)}_{\text{prior distribution}} \quad (2.1a)$$

$$P(M|E) = \frac{P(E|M)P(M)}{P(E)} \quad (2.1b)$$

Here, the prior probability  $P(M)$ , represents what we know or believe about the function's properties *a priori*. The posterior probability  $P(M|E)$  then represents our updated beliefs about the function on observing further evidence. The equality [2.1b](#) can be proven by inspection using the expression for the conditional probability of the model  $M$  given the evidence  $E$ , as in [2.2](#).

$$P(M|E) = \frac{P(M \cap E)}{P(E)} = \frac{P(M \cap E)}{P(E)} \frac{P(M)}{P(M)} = \frac{P(E \cap M)}{P(M)} \frac{P(M)}{P(E)} = \frac{P(E|M)P(M)}{P(E)} \quad (2.2)$$

Gaussian Processes can be used to represent the prior model  $M$ , where the *training data* (the training inputs and observed outputs) represent the evidence  $E$  and the GP predictions represent the posterior distribution  $P(M|E)$ . *Nonlinear regression* aims to model an unknown continuous function from a given *training set* of input-output pairs. *Parametric* regression techniques attempt to infer a finite number of parameters from the training set, which when applied to a given function class, best describe the input-output data e.g. linear regression. In *nonparametric* estimation, the unknown function  $f$  is not specified explicitly using parameter values. Instead, the function estimate is formed directly based on (noisy) observations of function values and regularity properties. *Bayesian nonparametric models* are defined on an infinite-dimensional parameter space e.g. the possible set of continuous functions in the case of nonlinear regression. This approach

defines a prior distribution over possible continuous functions and predicts a posterior distribution for a specified set of inputs given the observed training data using *Gaussian Processes* [37]. The Gaussian Process regression framework is used in this work to model unknown functions in real-time at a low computational expense and with a low sampling rate to a degree of accuracy sufficient for use in a MPC control scheme.

## 2.2 Gaussian Distributions

**Definition 2.2.1** (Gaussian Distribution, [38] Appendix A.2). *A multivariate Gaussian distribution is the joint probability of a finite number of free variables.*

The probability density function (see Appendix [B.1] and [B.4]) of a Gaussian (or *univariate/multivariate normal*) distribution is expressed by [2.3a], in the compact form by [2.3b] and is illustrated by Fig. [2.1]

$$p(\mathbf{x}|\boldsymbol{\mu}, K) = (2\pi)^{-\frac{n_x}{2}} |K|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T K^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \quad (2.3a)$$

$$\mathbf{x}|\boldsymbol{\mu}, K \sim \mathcal{N}(\boldsymbol{\mu}, K) \quad (2.3b)$$

where:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_x} \end{bmatrix} \in \mathbb{R}^{n_x} \text{ is the vector of random variables}$$

$$\boldsymbol{\mu} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n_x} \end{bmatrix} \in \mathbb{R}^{n_x} \text{ is the vector of means corresponding to each dimension of } \mathbf{x}$$

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_{n_x}) \\ k(x_2, x_1) & \ddots & & \vdots \\ \vdots & & \ddots & \\ k(x_{n_x}, x_1) & \cdots & & k(x_{n_x}, x_{n_x}) \end{bmatrix} \in \mathbb{R}^{n_x \times n_x} \text{ is the covariance matrix}$$

where element  $k(x_i, x_j)$  specifies the joint variability between random variables  $x_i$  and  $x_j$ .

The *joint Gaussian distribution* (see Appendix [B.2] and [B.4]) of two vectors of (possibly interdependent) random variables  $\mathbf{x}$  and  $\mathbf{y}$ , or the probability of sampling particular vectors of  $\mathbf{x}$  and  $\mathbf{y}$ , is given in compact form by [2.4]

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} K_{xx} & K_{xy} \\ K_{yx} & K_{yy} \end{bmatrix}\right) \quad (2.4)$$

where:

$\boldsymbol{\mu}_a$  are the mean values of the random variables  $\mathbf{a}$

$K_{ab}$  is the covariance matrix specifying the variability between random variables  $\mathbf{a}$  and  $\mathbf{b}$

The joint distribution [2.4] can be *marginalized* over  $\mathbf{y}$  to give the *marginal distribution* (see Appendix [B.2] and [B.4]) of  $\mathbf{x}$ , i.e. the probability of sampling a particular vector  $\mathbf{x}$  over all possible vectors  $\mathbf{y}$ , and is given in compact form by [2.5]

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, K_{xx}) \quad (2.5)$$

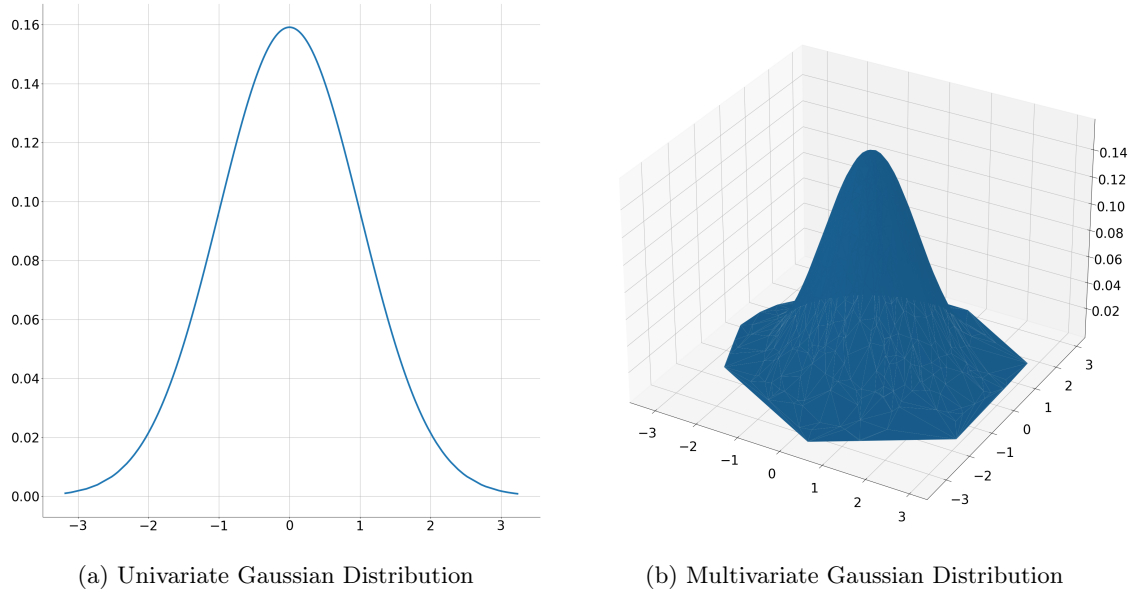


Figure 2.1: The univariate Gaussian distribution in Fig. 2.1a is a function of a single random variable, whereas the multivariate Gaussian distribution in Fig. 2.1b is a function of two interdependent random variables.

The *conditional distribution* (see Appendix B.2 and B.4) of  $\mathbf{x}$  given  $\mathbf{y}$  is given in compact form by 2.6

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}} + K_{xy}K_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}), K_{xx} - K_{xy}K_{yy}^{-1}K_{xy}^T) \quad (2.6)$$

## 2.3 Gaussian Processes

**Definition 2.3.1** (Gaussian Process, [38] Def. 2.1). *A Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A GP is completely specified by its *mean function*  $m(\mathbf{x})$  (2.7) and *covariance function*  $k(\mathbf{x}, \mathbf{x}')$  (2.8).

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \quad (2.7)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \quad (2.8)$$

A function  $f(\mathbf{x})$  represented by a GP can be expressed as a *distribution over possible functions*, denoted by 2.9.

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (2.9)$$

The function values at any finite number of inputs, e.g.  $f(\mathbf{x}_1)$  and  $f(\mathbf{x}_2)$ , have a joint Gaussian distribution (2.10), in that the function value for each random vector  $\mathbf{x}_i$  has a corresponding mean  $m(\mathbf{x}_i)$  and the interdependence of the function values are expressed by their covariance matrix entries given by the function  $k$ .

$$\begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} m(\mathbf{x}_1) \\ m(\mathbf{x}_2) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) \end{bmatrix} \right) \quad (2.10)$$

GPs exhibit a *marginalization property* (or a *consistency requirement*). This means that if the GP specifies a joint distribution then it can also specify that distribution marginalized

over one or more of the inputs. For example, the joint probability distribution in [2.10](#) can be marginalized over  $f(\mathbf{x}_2)$  to give the distribution of the function value  $f(\mathbf{x}_1)$  in [2.11](#).

$$f(\mathbf{x}_1) \sim \mathcal{N}(m(\mathbf{x}_1), k(\mathbf{x}_1, \mathbf{x}_1)) \quad (2.11)$$

Intuitively, the *random variables*  $[f(\mathbf{x}_1), f(\mathbf{x}_2)]$  can be thought of as the values of the unknown function  $f$  at inputs  $[\mathbf{x}_1, \mathbf{x}_2]$  from the continuous function domain. The function value at any given input,  $f(\mathbf{x})$ , is uncertain and has a Gaussian distribution given by [2.12](#).

$$f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x})) \quad (2.12)$$

The GP is therefore the stochastic analogy to the deterministic (or true) scalar function. Whereas the deterministic function returns a scalar  $f(\mathbf{x})$  for an arbitrary input  $\mathbf{x}$ , the stochastic GP returns the mean and variance of a normal distribution over the possible values of  $f$  at  $\mathbf{x}$  (see [38](#) Sec 2.2 for an in-depth description of this *function-space view* of Gaussian Processes).

## 2.4 Covariance Function

The choice of mean function [\(2.7\)](#) and the covariance function [\(2.8\)](#) for a Gaussian Process make it possible to encode prior beliefs about the properties of the unknown function into the approximation. Whereas the mean function is generally taken to have a constant zero value (see [38](#) Sec. 2.2), the covariance function plays a significant role in the accuracy of the GP model.

The *covariance function* or *kernel* specifies the covariance between pairs of random variables and is defined in Def. [2.4.1](#).

**Definition 2.4.1** (Kernel, [39](#)). *A function  $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$  is a kernel if:*

1.  *$k$  is symmetric:  $k(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_2, \mathbf{x}_1)$*
2.  *$k$  is positive semi-definite:  $\forall \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathcal{X}$ , the Gram Matrix  $K \in \mathbb{R}^{n \times n}$  defined by  $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$  is positive semi-definite (i.e.  $\forall \mathbf{v} \in \mathbb{R}^n, \mathbf{v}^T \mathbf{v} \geq 0$ ).*

For the purposes of modeling functions as Gaussian Processes, the covariance between the function values at two different inputs,  $f(\mathbf{x}_1)$  and  $f(\mathbf{x}_2)$ , is expressed as a function of the two corresponding input random variables,  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , see [2.13](#).

$$\text{cov}(f(\mathbf{x}_1), f(\mathbf{x}_2)) := k(\mathbf{x}_1, \mathbf{x}_2) \quad (2.13)$$

This is based on the understanding that points along the function  $f$  with inputs  $\mathbf{x}_1$  and  $\mathbf{x}_2$  that are close (as measured by the induced vector norm of the topological space) to each other (and thus have a high covariance) are likely to have observed function values (or *target* values),  $y = f(\mathbf{x}) + \kappa$ , which are close to each other.

An arbitrary function of input vectors  $\mathbf{x}_1$  and  $\mathbf{x}_2$  will not necessarily be a valid covariance kernel. There are, however, some commonly-used models with well-established properties. A *stationary* covariance kernel is a function of  $\mathbf{x}_1 - \mathbf{x}_2$  and is therefore invariant to translations in the input space  $\mathcal{X}$ . One issue that arises with stationary kernels is that they do not consider function classes with different rates of change w.r.t. the inputs to be likely. An *isotropic* kernel is, in addition, a function only of the *radius*  $r = |\mathbf{x}_1 - \mathbf{x}_2|$  and is therefore invariant to all rigid motions. Isotropic kernels are also referred to as *radial basis functions* (RBF), as they are a function of the radius only (see [38](#) Chapter 4 for a comprehensive description of different covariance kernels).

The *squared exponential* (SE) covariance kernel has been successfully implemented in literature [13](#), [24](#) and is given by Eqn. [2.14](#).

$$k_{SE}(\mathbf{x}_1, \mathbf{x}_2) = \sigma^2 \exp\left(-\frac{1}{2}(\mathbf{x}_1 - \mathbf{x}_2)^T \text{diag}(\mathbf{1})^{-2}(\mathbf{x}_1 - \mathbf{x}_2)\right) + \sigma_n^2 \delta_{12} \quad (2.14)$$

where:

$\mathbf{l} \in \mathbb{R}_{++}^{n_x}$  is the *characteristic length scale*

$\sigma$  is the *output variance*

$\sigma_n$  is the *measurement noise*

$\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{n_x}$  are two arbitrary input vectors

The SE covariance function is infinitely differentiable, meaning that it has mean square derivatives (see Appendix [B.7](#)) of all orders and is thus very smooth. It is also infinitely divisible, meaning that  $(k(\mathbf{x}_1, \mathbf{x}_2))^t$  is a valid kernel (i.e. symmetric and positive semi-definite) for all  $t > 0$ .

The so-called *hyper-parameters* of the SE covariance kernel  $\sigma$  and  $\mathbf{l}$  need to be tuned correctly such that the Gaussian Process will approximate the unknown function to a sufficient degree of accuracy without over-fitting. Note that the length scale is a vector, and can be tuned to vary for each dimension of the input  $\mathbf{x}$ . When multiple Gaussian Processes are used to model a collection of unknown functions, the optimal parameters are naturally different for each case.

## 2.5 Hyperparameter Selection

In order for the Gaussian Process model to be considered useful, it must be specified in a way that will most accurately represent the true function. In our case, we must specify the prior mean of the function and the prior covariance. The mean function can be set to a linearized approximation of the function, to any number of known terms in the function or, most frequently, to a zero constant. Generally the covariance function is more involved and is specified by both the discrete choice of kernel function and the continuous setting of its hyperparameters. *Training* of a Gaussian Process refers to both the selection of the covariance function and its parameters (see [38](#) Chapter 5).

The SE covariance kernel used for each GP in this work has a hyperparameter vector  $\boldsymbol{\theta} = [\sigma, \mathbf{l}^T]^T$  (the standard deviation of the output and the length scales, respectively). Notice that we take the standard deviation of the output (as opposed to the variance) such that we can optimize for these values over the full domain of  $\mathbb{R}/\{0\}$ , which simplifies the hyperparameter selection process described later in this section.

In some cases the meaning of the hyperparameters is intuitive, which aids in the selection process and also in interpretation of the training data given the optimized hyperparameters. For example, the characteristic length scale for a given dimension  $i$ ,  $l_i$ , gives the approximate distance along dimension  $i$  in the input space between two uncorrelated function values i.e. an adjacent peak and trough in the function along axis  $i$ . Loosely speaking, the variation in the observed outputs along that axis which are *less* than the corresponding length scale will be neglected by the GP prediction. The SE function thus implements *automatic relevance determination* (ARD) (see [40](#)) in that the inverse of the length-scale specifies how relevant variations along that dimension are, and thus how relevant that feature is to the covariance. This effect can be used to effectively exclude the influence of certain inputs from the Gaussian Process regression by imposing a very large length scale along the corresponding dimension.

Two prominent approaches to finding suitable parameters include *cross-validation* and *maximizing the marginal log likelihood* of the observed outputs given the training data and a particular choice of hyperparameters (see [38](#) Sec. 5.4.1 for a comparison of different model and hyperparameter selection techniques). We will apply the latter approach, in which Bayesian inference is applied to express the distribution of the observed outputs given the hyperparameters. In the case of Gaussian Process regression, the integrals over the hyperparameters in the marginalized distribution are analytically tractable (see [38](#) Sec. 5.4.1). The *marginal log likelihood* of seeing the observed outputs  $\mathbf{y}$  *conditioned* on the true function values  $\mathbf{f}$ , the training data  $X$  and the hyperparameters  $\boldsymbol{\theta}$  and subsequently *marginalized* over the true function values is given by. Eqn.



**2.15**

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \underbrace{-\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y}}_{\text{data-fit term}} - \underbrace{\frac{1}{2}\log |K_y|}_{\text{complexity penalty}} - \underbrace{\frac{N_{tr}}{2}\log 2\pi}_{\text{normalization constant}} \quad (2.15)$$

where:

$K_y = K_f + \sigma_n^2 I$  is the covariance matrix for the noisy observations  $\mathbf{y}$

$K_f = K(X, X)$  is the covariance matrix for the noise-free observations  $f$

In seeking the optimal hyperparameters  $\boldsymbol{\theta}^*$ , we aim to maximize the marginal log likelihood with respect to  $\boldsymbol{\theta}$  in the unconstrained optimal control problem **2.16**

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^{1+n_x}} \log p(\mathbf{y}|X, \boldsymbol{\theta}) \quad (2.16)$$

The solution to **2.16** can be found given the expression **2.17** for the partial derivatives of the marginal log likelihood w.r.t.  $\boldsymbol{\theta}$ .

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2} \text{tr} \left( (\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K_y^{-1}) \frac{\partial K_y}{\partial \theta_j} \right) \quad (2.17)$$

where  $\boldsymbol{\alpha} = K_y^{-1}\mathbf{y}$ .

The optimal hyperparameters  $\boldsymbol{\theta}^*$  can thus be found using a gradient-based optimization method, since the time-complexity of calculating the derivatives **2.17** is small ( $\mathcal{O}(N_{tr}^2)$  per hyperparameter  $\theta_j$ ) once the more computationally expensive inverted covariance matrix  $K_y^{-1}$  has been computed ( $\mathcal{O}(N_{tr}^3)$  for matrix inversion of symmetric positive-definite matrices).

## 2.6 Bayes' Theorem & Gaussian Processes

Gaussian Processes (Def. **2.3.1**) are a tractable realization of Bayesian nonparametric nonlinear regression. Prior beliefs about the function are encoded in mean and covariance functions (illustrated by Fig. **2.2a** for a prior mean of 0 and a SE covariance kernel) and combined with observed (noise-free or noisy) function outputs to generate posterior mean and covariance values (illustrated for increasing training set size by Fig. **2.2b** and **2.2c**).

Let  $X$  be the  $N_{tr} \times n_x$  matrix of *training points*,  $\mathbf{y}$  be the  $N_{tr} \times 1$  vector of observed outputs (or *target values*),  $X_*$  be the  $N_* \times n_x$  matrix of *test points* and  $\mathbf{f}_*$  be the  $N_* \times 1$  vector of *test outputs* we wish to predict for each point (row) in  $X_*$ . Furthermore, let  $\boldsymbol{\mu}_y$  and  $\boldsymbol{\mu}_{f_*}$  be the mean vectors corresponding to  $\mathbf{y}$  and  $\mathbf{f}_*$ , respectively; and  $K$  be the covariance matrix coupling these two Gaussian-distributed vectors. Assuming that both the noisy training outputs  $\mathbf{y}$  and the (yet unknown) test outputs  $\mathbf{f}_*$  are both described by Gaussian distributions, their joint prior distribution is given by **2.18**

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, K) \quad (2.18)$$

where:

$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_y \\ \boldsymbol{\mu}_{f_*} \end{bmatrix}$  is the vector of mean values

$K = \begin{bmatrix} K_{y,y} & K_{y,f_*} \\ K_{f_*,y} & K_{f_*,f_*} \end{bmatrix}$  is the covariance matrix

Recall from the introduction on GPs in Sec. 2.3 that the joint distribution of any finite collection of random variables from a GP is also a GP. To obtain the *posterior* distribution over the possible test outputs  $\mathbf{f}_*$ , we *condition* the joint prior distribution  $\mathbf{f}_*$  on the observations  $X, \mathbf{y}$  as well as on the given test inputs  $X_*$ , obtaining the distribution 2.19,

$$\mathbf{f}_* | X_*, X, \mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{f}_*} + K_{\mathbf{f}_*, \mathbf{y}} K_{\mathbf{y}, \mathbf{y}}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}), K_{\mathbf{f}_*, \mathbf{f}_*} - K_{\mathbf{f}_*, \mathbf{y}} K_{\mathbf{y}, \mathbf{y}}^{-1} K_{\mathbf{y}, \mathbf{f}_*}) \quad (2.19)$$

Expression 2.19 is derived from Bayes' Theorem (see 2.1) in that the posterior distribution ( $P(M|E) = \mathbf{f}_* | X_*, X, \mathbf{y}$ ) is the prior model ( $P(M) = \mathcal{N}(\boldsymbol{\mu}, K)$ ) conditioned on the observed function outputs ( $E = (X, \mathbf{y})$ ).

## 2.7 Predicting Outputs with Noise-Free Observations

The general expression for predicting outputs from a GP given an arbitrary prior mean vector  $\boldsymbol{\mu}$  and covariance matrix  $K$  is given by 2.19. In this section we will outline the GP prediction procedure for the specific case of noise-free target values.

Given that the function output observations are noise-free, we can assume that the training set  $(X, \mathbf{f}) = \{(\mathbf{x}_i, f_i) | i = 1, \dots, N_{tr}\}$  is known, where the observed outputs  $f_i$  are equivalent to the true function outputs  $f(\mathbf{x}_i)$ .

The joint distribution of the vector of previously observed outputs  $\mathbf{f}$ , and the vector of outputs  $\mathbf{f}_*$  we wish to predict at given test inputs  $X_*$ , is given by 2.20,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{f}} \\ \boldsymbol{\mu}_{\mathbf{f}_*} \end{bmatrix}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \quad (2.20)$$

The conditional distribution of the function,  $\mathbf{f}_*$  conditioned on observed training data,  $(X, \mathbf{f})$  is then given by 2.21,

$$\mathbf{f}_* | X_*, X, \mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}_*, K(\mathbf{f}_*, \mathbf{f}_*)) \quad (2.21)$$

where:

$\bar{\mathbf{f}}_* = \boldsymbol{\mu}_{\mathbf{f}_*} + K(X_*, X)K(X, X)^{-1}(\mathbf{f} - \boldsymbol{\mu}_{\mathbf{f}})$  is the posterior mean of the GP prediction

$K(\mathbf{f}_*, \mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)$  is the posterior covariance matrix of the GP prediction

## 2.8 Predicting Outputs with Noisy Observations

The prediction given by 2.21 is not applicable to most realistic cases in which the target values  $\mathbf{y}$  are not precisely reflective of the true function outputs  $\mathbf{f}$  but rather noisy samples of the function. In this section we will outline the GP prediction procedure for the case of noisy target values.

Consider the case where the function output observations have been corrupted by a Gaussian noise with mean 0 and variance  $\sigma_n^2$ . In this scenario, we assume that the noisy training set  $(X, \mathbf{y}) = \{(\mathbf{x}_i, y_i) | i = 1, \dots, N_{tr}\}$  is known, where the observed outputs  $y_i$  are noisy observations of the true function outputs  $f(\mathbf{x}_i)$ .

The joint distribution of the vector of previously observed outputs  $\mathbf{y}$ , and the vector of outputs  $\mathbf{f}_*$  we wish to predict at given test inputs  $X_*$ , is given by 2.22,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{y}} \\ \boldsymbol{\mu}_{\mathbf{f}_*} \end{bmatrix}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \quad (2.22)$$

The conditional distribution of the function  $\mathbf{f}_*$  conditioned on observed training data  $(X, \mathbf{y})$  is then given by [2.23](#):

$$\mathbf{f}_* | X_*, X, \mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}_*, K(\mathbf{f}_*, \mathbf{f}_*)) \quad (2.23)$$

where:

$\bar{\mathbf{f}}_* = \boldsymbol{\mu}_{\mathbf{f}_*} + K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}(\mathbf{f} - \boldsymbol{\mu}_{\mathbf{f}})$  is the posterior mean of the GP prediction

$K(\mathbf{f}_*, \mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*)$  is the posterior covariance matrix of the GP prediction

## 2.9 Measuring the Performance of a GP Prediction

The performance of a GP approximation can be measured by comparing the true values of the function,  $f_i$ , and the predicted posterior mean values of the function,  $\bar{f}(\mathbf{x}_i)$ , at a number of test points,  $X = \{\mathbf{x}_i | i = 1, \dots, N_*\}$ . This is calculated based on the ratio between the Residual Sum of Squares (RSS) and the Total Sum of Squares (TSS), as in Eqn. [2.24](#).

$$\text{Score}_{\text{GP}} = \begin{cases} 1 - \frac{\text{RSS}}{\text{TSS}} & \text{if RSS} > 0 \text{ and TSS} > 0 \\ 1 & \text{if RSS} = \text{TSS} = 0 \\ 0 & \text{if RSS} > 0 \text{ and TSS} = 0 \end{cases} \quad (2.24)$$

where:

$\text{RSS} = \sum_{i=1}^{N_*} (f_i - \bar{f}(\mathbf{x}_i))^2$  is the Residual Sum of Squares (RSS)

$\text{TSS} = \sum_{i=1}^{N_*} (f_i - \mathbb{E}(\mathbf{f}))^2$  is the Total Sum of Squares (TSS)

$\mathbb{E}(\mathbf{f})$  is the mean of all true function values  $\mathbf{f} = \{f(\mathbf{x}_i) | i = 1, \dots, N_*\}$

## 2.10 Computing the First-Order Derivative of the Posterior Mean

For the purposes of implementing GP-modeled functions in an optimal control problem using a gradient-based optimization method, we will need to compute the first-order derivative of the posterior mean at an arbitrary test-point  $\mathbf{x}_*$ ,  $\nabla \bar{f}(\mathbf{x}_*)$ . In this work, two approaches are utilized: the finite-difference method and the closed-form expression for the first-order derivative of the posterior mean (see [41](#)), where the former is used to validate the latter.

### 2.10.1 Finite-Difference Method

The gradient of the posterior mean of a GP, for any covariance kernel, can be calculated with the finite-difference method as in Eqn. [2.25](#)

$$\nabla \bar{f}(\mathbf{x}_*) = \frac{1}{2\delta} \begin{bmatrix} \bar{f}(\mathbf{x}_* + [\delta, 0, \dots, 0]^T) - \bar{f}(\mathbf{x}_* - [\delta, 0, \dots, 0]^T) \\ \bar{f}(\mathbf{x}_* + [0, \delta, \dots, 0]^T) - \bar{f}(\mathbf{x}_* - [0, \delta, \dots, 0]^T) \\ \vdots \\ \bar{f}(\mathbf{x}_* + [0, 0, \dots, \delta]^T) - \bar{f}(\mathbf{x}_* - [0, 0, \dots, \delta]^T) \end{bmatrix} \quad (2.25)$$

where  $\delta$  is an incremental difference constant.

### 2.10.2 Closed-Form First-Order Derivative of the Posterior Mean

The gradient of the posterior mean of a GP using a SE kernel can be analytically calculated with Eqn. [2.26](#).

$$\nabla \bar{f}(\mathbf{x}_*) = -\text{diag}(\mathbf{I}^2)^{-1}[\mathbf{x}_* - \mathbf{x}_1, \dots, \mathbf{x}_* - \mathbf{x}_{N_{\text{tr}}}] (K(X, \mathbf{x}_*) \odot K(X, X)^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)) \quad (2.26)$$

where:

$\odot$  represents an element-wise product

$\mathbf{x}_i$  is the  $i^{\text{th}}$  point (row) of the input training data  $X$

$\mathbf{y}$  is the mean of the observed target values

## 2.11 Reproducing Kernel Hilbert Spaces

Reproducing kernel Hilbert spaces (RKHS) define a Hilbert space (see Def. [2.11.2](#)) of sufficiently-smooth functions corresponding to a given positive semi-definite kernel  $k$  (see [38](#) Sec. 6.1). In this section we will provide definitions key to understanding these function spaces and how they relate to Gaussian Processes.

**Definition 2.11.1** (Inner Product). *An inner product  $\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{F}$  maps the vector space  $\mathcal{X}$  to the scalar space  $\mathcal{F}$  and satisfies the following conditions:*

1. *Conjugate Symmetry:*

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \overline{\langle \mathbf{x}_2, \mathbf{x}_1 \rangle} \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X} \quad (2.27)$$

2. *Bilinearity:*

$$\langle \alpha \mathbf{x}_1 + \beta \mathbf{x}_2, \mathbf{x}_3 \rangle = \alpha \langle \mathbf{x}_1, \mathbf{x}_3 \rangle + \beta \langle \mathbf{x}_2, \mathbf{x}_3 \rangle \quad \forall \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{X}, \forall \alpha, \beta \in \mathbb{R} \quad (2.28)$$

3. *Positive Definiteness:*

$$\begin{aligned} \langle \mathbf{x}, \mathbf{x} \rangle &\geq 0 \quad \forall \mathbf{x} \in \mathcal{X} \\ \langle \mathbf{x}, \mathbf{x} \rangle &= 0 \Leftrightarrow \mathbf{x} = \mathbf{0} \end{aligned} \quad (2.29)$$

**Definition 2.11.2** (Hilbert Space [39](#)). *A Hilbert space  $\mathcal{H}$  is an inner product space (i.e. a space of real functions  $f$  defined on an index set  $\mathcal{X}$  and endowed with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  that is complete and separable with respect to the norm defined by the inner product  $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$ .*

**Definition 2.11.3** (Reproducing Kernel). *The function  $k(\cdot, \cdot)$  is a reproducing kernel of a Hilbert space  $\mathcal{H}$  if  $\forall f \in \mathcal{H}, f(\mathbf{x}) = \langle k(\mathbf{x}, \cdot), f(\cdot) \rangle_{\mathcal{H}}$*

**Definition 2.11.4** (Reproducing Kernel Hilbert Space [39](#)). *A reproducing kernel Hilbert space (RKHS) is a Hilbert space  $\mathcal{H}_k$  with a reproducing kernel  $k$  whose span is dense in  $\mathcal{H}_k$  (i.e. every function  $f \in \mathcal{H}_k$  is either in  $\text{span}(k)$  or is a limit point of  $\text{span}(k)$ ). Equivalently, a RKHS can be defined as a Hilbert space of functions with all evaluation functionals (i.e. all possible function outputs) bounded and linear.*

Theorem [2.11.5](#) tells us that a RKHS  $\mathcal{H}_k$  uniquely determines a kernel  $k$ , and vice-versa.

**Theorem 2.11.5** (Moore-Aronszajn Theorem, [42](#)). *Let  $\mathcal{X}$  be an index set. Then for every positive definite function  $k(\cdot, \cdot)$  on  $\mathcal{X} \times \mathcal{X}$  there exists a unique RKHS, and vice-versa.*

Let  $f(\mathbf{x}) = \sum_{i=1}^N f_i \phi_i(\mathbf{x})$  be the eigenexpansion of the function  $f$ . If we sample the coefficients  $f_i$  from a Gaussian distribution  $\mathcal{N}(0, \lambda_i)$  and  $N$  is infinite, we can consider that the infinite collection of random variables  $f_1, f_2, \dots$  is a Gaussian Process with zero mean and variances  $\lambda_1, \lambda_2, \dots$ . Although the sample functions of  $f$  are not in the RKHS  $\mathcal{H}_k$  (unbounded functional evaluation for  $N = \infty$ ), the posterior mean of  $f$  given some observed data  $E$  will lie in the RKHS. A function in a RKHS can thus be modeled by a GP.

## 2.12 Gaussian Processes for Dynamic State Functions

This section outlines the mathematical framework required to model a number of dynamic state functions as GPs.

Consider a collection of  $D$  arbitrary controllable devices, where the state dynamics of each controllable device  $d$  depends on the  $n_x^d$  current states,  $n_u^d$  control inputs and  $n_w^d$  exogenous disturbances. This relationship can be modeled by a number of Gaussian Processes, one for each state  $i = 1, \dots, n_x^d$ . The dynamics of each state  $i$  of a single device  $d$  can thus be estimated by forming a prior of the state, building a training set of noisy input samples to, and output samples from, the true system and calculating the posterior distribution of the state given the training data. The *known function*  $h_i^d$  represents the known or assumed terms in the dynamic state function while the *unknown nonlinear function*  $g_i^d$  represents the black-box terms. With this formulation, we can employ Gaussian Processes to only estimate the unknown contributions to the dynamic state function as outlined in [2.30](#).

When a GP-modeled dynamic state function is employed by a MPC scheme, it is used to predict the behavior of the system over a finite number of future time-steps. Recall that the output of the GP is a *distribution* rather than a scalar value. Therefore, if the system states resulting from GP predictions are recursively fed to the GP model beyond the first time-step, the predicted system outputs will become progressively more complex as the uncertainty (as measured by the prediction variance) is propagated through the horizon. The *zero-variance method*, in which the posterior mean values of the GPs are used to predict system behavior over the horizon, does not propagate uncertainty and was shown in [\[43\]](#) to achieve sufficient prediction accuracy when compared to the alternative Monte-Carlo method of uncertainty propagation. The zero-variance method was therefore chosen to simulate the response of the system in the MPC scheme proposed in this work.

$$x_{k+1,i}^d := f_i^d(\mathbf{z}_k^d) = \underbrace{h_i^d(\mathbf{z}_k^d)}_{\text{known state prior}} + \underbrace{g_i^d(\mathbf{z}_k^d)}_{\text{unknown state variation}} \quad (2.30a)$$

$$\text{Measurement Gaussian Noise, } \kappa \sim \mathcal{N}(0, \sigma_{n,g_i^d}^2) \quad (2.30b)$$

$$\text{Noisy Measured State, } \tilde{x}_{k,i}^d := x_{k,i}^d + \kappa \quad (2.30c)$$

$$\text{Noisy Measured State Vector, } \tilde{\mathbf{x}}_{k,i}^d \in \mathbb{R}^{n_x^d} := [\tilde{x}_{k,i}^d \mid i = 1, \dots, n_x^d]^T \quad (2.30d)$$

$$\text{Input Training Data Point, } \mathbf{z}_k^d \in \mathbb{R}^{n_x^d + n_u^d + n_w^d} := [\tilde{\mathbf{x}}_k^{dT}, \mathbf{u}_k^{dT}, \mathbf{w}_k^{dT}]^T \quad (2.30e)$$

$$\text{Input Training Data Set, } Z^d \in \mathbb{R}^{N_{tr} \times (n_x^d + n_u^d + n_w^d)} := \{\mathbf{z}_t^d \mid t = 1, \dots, N_{tr}\} \quad (2.30f)$$

$$\text{Output Training Data Point, } \tilde{g}_{k,i}^d := \tilde{x}_{k,i}^d - h_i^d(\mathbf{z}_k^d) \quad (2.30g)$$

$$\text{Output Training Data Set, } \tilde{\mathbf{g}}_i^d := \{\tilde{g}_{i,t}^d \mid t = 1, \dots, N_{tr}\} \quad (2.30h)$$

$$\text{Training Data Set, } (Z^d, \tilde{\mathbf{g}}_i^d) \quad (2.30i)$$

$$\text{Input Test Data Point, } \mathbf{z}_{k*}^d := [\tilde{\mathbf{x}}_{k*}^{dT}, \mathbf{u}_{k*}^{dT}, \mathbf{w}_{k*}^{dT}]^T \quad (2.30j)$$

$$\text{Input Test Data Set, } Z_*^d := \{\mathbf{z}_{t,*}^d \mid t = 1, \dots, N_*\} \quad (2.30k)$$

Joint Gaussian Probability Distribution,

$$\begin{bmatrix} \mathbf{g}_{i*}^d \\ \mathbf{g}_{i*}^d \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} K(Z^d, Z^d) + \sigma_{n,g_i^d}^2 I & K(Z^d, Z_*^d) \\ K(Z_*^d, Z^d) & K(Z_*^d, Z_*^d) \end{bmatrix} \right) \quad (2.30l)$$

(Marginalized) Posterior Gaussian Distribution,

$$\mathbf{g}_{i*}^d \mid Z^d, \mathbf{g}_i^d, Z_*^d \sim \mathcal{N}(\bar{\mathbf{g}}_{i*}^d, K(\mathbf{g}_{i*}^d, \mathbf{g}_{i*}^d)) \quad (2.30m)$$

Posterior Mean,

$$\bar{\mathbf{g}}_{i*}^d := \mathbb{E}[g_{i*}^d \mid Z^d, g_i^d, Z_*^d] = K(Z_*^d, Z^d)[K(Z^d, Z^d) + \sigma_{n,g_i^d}^2 I]^{-1} \mathbf{g}_i^d \quad (2.30n)$$

Posterior Variance,

$$K(\mathbf{g}_{i*}^d, \mathbf{g}_{i*}^d) := K(Z_*^d, Z_*^d) - K(Z_*^d, Z^d)[K(Z^d, Z^d) + \sigma_{n,g_i^d}^2 I]^{-1} K(Z^d, Z_*^d) \quad (2.30o)$$

where  $\sigma_{n,g_i^d}^2$  is the variance associated with the state feedback measurements of device  $d$ , state  $i$  and  $k$  is the absolute time-step in the simulation.

## 2.13 Gaussian Processes for Cost Functions

This section outlines the mathematical framework required to model a number of cost functions as GPs.

Again, consider a collection of  $D$  arbitrary controllable devices, where the cost to the user (or the *user dissatisfaction function*) of each device depends on the  $n_x^d$  current states,  $n_u^d$  control inputs and  $n_w^d$  exogenous disturbances. Similar to the dynamic state case, this relationship can be modeled by a single Gaussian Process for each device. The cost to the user of a single device  $d$  can thus be estimated by forming a prior of the cost function, building a training set of inputs and perceived cost samples from user feedback, and calculating the posterior distribution given the training data. The *known function*  $m^d$  represents the known terms in the cost function while the *unknown function*  $j^d$  represents the black-box function terms. With this formulation, we can employ the Gaussian Processes to estimate unknown contributions to the a stage cost function which is representative of user dissatisfaction, as outlined by [2.31](#).

$$l_k^d := l^d(\mathbf{z}_k^d) = \underbrace{m^d(\mathbf{z}_k^d)}_{\text{known stage cost prior}} + \underbrace{j^d(\mathbf{z}_k^d)}_{\text{unknown stage cost variation}} \quad (2.31a)$$

$$\text{Input Training Data Point, } \mathbf{z}_k^d \in \mathbb{R}^{n_x+n_u+n_w} := [\tilde{\mathbf{x}}_k^{dT}, \mathbf{u}_k^{dT}, \mathbf{w}_k^{dT}]^T \quad (2.31b)$$

$$\text{Input Training Data Set, } Z^d \in \mathbb{R}^{N_{tr} \times (n_x+n_u+n_w)} := \{\mathbf{z}_t^d \mid t = 1, \dots, N_{tr}\} \quad (2.31c)$$

$$\text{Measurement Gaussian Noise, } \kappa \sim \mathcal{N}(0, \sigma_{n,j^d}^2) \quad (2.31d)$$

$$\text{Noisy Measured Cost, } \tilde{l}_k^d := l_k^d + \kappa \quad (2.31e)$$

$$\text{Output Training Data Point, } \tilde{j}_k^d := \tilde{l}_k^d - m^d(\mathbf{z}_k^d) \quad (2.31f)$$

$$\text{Output Training Data Set, } \tilde{\mathbf{j}}^d := \{\tilde{j}_t^d \mid t = 1, \dots, N_{tr}\} \quad (2.31g)$$

$$\text{Training Data Set, } (Z^d, \tilde{\mathbf{j}}^d) \quad (2.31h)$$

$$\text{Input Test Data Point, } \mathbf{z}_{k*}^d := [\tilde{\mathbf{x}}_{k*}^{dT}, \mathbf{u}_{k*}^{dT}, \mathbf{w}_{k*}^{dT}]^T \quad (2.31i)$$

$$\text{Input Test Data Set, } Z_*^d := \{\mathbf{z}_{t,*}^d \mid t = 1, \dots, N_*\} \quad (2.31j)$$

Joint Gaussian Probability Distribution,

$$\begin{bmatrix} \mathbf{j}^d \\ \mathbf{j}_*^d \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(Z^d, Z^d) + \sigma_{n,j^d}^2 I & K(Z^d, Z_*^d) \\ K(Z_*^d, Z^d) & K(Z_*^d, Z_*^d) \end{bmatrix}\right) \quad (2.31k)$$

(Marginalized) Posterior Gaussian Distribution,

$$\mathbf{j}_*^d | Z^d, \mathbf{l}^d, Z_*^d \sim \mathcal{N}(\tilde{\mathbf{j}}_*^d, K(\mathbf{j}_*^d, \mathbf{j}_*^d)) \quad (2.31l)$$

Posterior Mean,

$$\tilde{\mathbf{j}}_*^d := \mathbb{E}[j_*^d | Z^d, \mathbf{l}^d, Z_*^d] = K(Z_*^d, Z^d)[K(Z^d, Z^d) + \sigma_{n,j^d}^2 I]^{-1} \mathbf{j}^d \quad (2.31m)$$

Posterior Variance,

$$K(\mathbf{j}_*^d, \mathbf{j}_*^d) := K(Z_*^d, Z_*^d) - K(Z_*^d, Z^d)[K(Z^d, Z^d) + \sigma_{n,j^d}^2 I]^{-1} K(Z^d, Z_*^d) \quad (2.31n)$$

where  $\sigma_{n,j^d}^2$  is the variance associated with the cost measurements of user  $d$  and  $k$  is the absolute time-step in the simulation.

## 2.14 Integrating GP Models with the MPC Scheme

There are a few possible approaches to defining the known state prior  $h_i^d$ . One approach (and the one adopted in this work) is to set it to 0 and to use the GP to estimate the entire function. If an approximated physics-based model is available, then a second approach is to set the prior to a linearized version of the dynamic system model with assumed parameters or to an established nonlinear synthetic model. A third approach, is to set the prior to the current state and to use the GP to estimate the change in the state i.e.  $h_i^d(\mathbf{z}_k^d) := x_{i,k}^d$  and  $g_i^d(\mathbf{z}_k^d)$  models  $x_{i,k+1}^d - x_{i,k}^d$ .

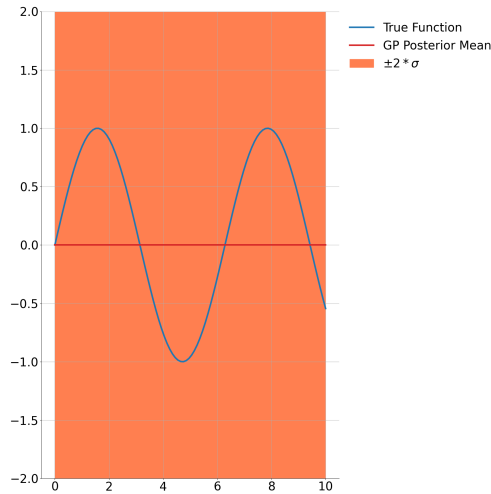
There are also several approaches to setting the prior over the cost functions. In this work, we set the prior to 0 and use the GP to estimate the entire function i.e.  $m^d(\mathbf{z}_{k_0}^d) := 0$  and  $j^d(\mathbf{z}_{k_0}^d)$  models  $l^d$ .

Care should be taken in defining how the training data is collected and how the covariance matrix  $K(X, X) + \sigma_n^2 I$  is inverted. The majority of the computational burden required to make predictions with a GP is involved in inverting this matrix, which can be minimized by limiting the number of training samples and by limiting how frequently the inversion must take place, or by exploiting sparsity properties of the covariance matrix. One approach is to collect system/user feedback data online until a pre-defined number of samples has been reached. The advantage of such an approach is that once all of the data has been collected, the covariance matrix can be inverted once, stored, and used as needed. Another approach (and the one adopted in this work)

is to initially train the GPs offline with  $N_{tr,0}$  training points, and to subsequently train the GPs with new system/user feedback in real-time up to a maximum of  $N_{tr}$  training points. A challenge to address in this approach is how to select new samples and how to discard existing samples once  $N_{tr}$  has been reached. In this work, a new feedback sample is only added to the training set if the standard deviation of the prediction at that point exceeds a pre-defined threshold, and the oldest sample is discarded to make space for it. The advantage of this approach is that the training set may reflect recent trends in the input-output data, but the drawback is that less-recent samples which contributed to the accuracy of the model in relatively unexplored input regions may be discarded. When the training data is augmented online like this, the covariance matrix is inverted and stored until the next datum of system/user feedback is received. This approach facilitates an *adaptive* control scheme in which the dynamic state models are initially defined with a given minimum number of training samples  $N_{tr,0}$  and are subsequently adapted with a sliding window set of  $N_{tr} - N_{tr,0}$  training samples as the control scheme is executed. The effect of this sliding-window online subset of training data is to fine-tune the accuracy of a GP over a particular input region as the control trajectory moves towards and through that region.

The GP model is of limited use given test values which deviate greatly from the available training data. For this reason we set *dynamic constraints* on the states and inputs of the MPC problem developed in Chapter 3, such that the states and inputs may not take on values outside of the upper and lower bounds observed in their respective dimensions of the training dataset. This does not guarantee that the feasible sets are subsets of the training dataset but that at least some of the input variables intersect with the training dataset.





(a) Prior Distribution over Functions

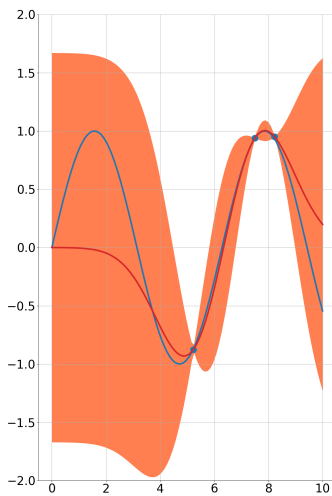
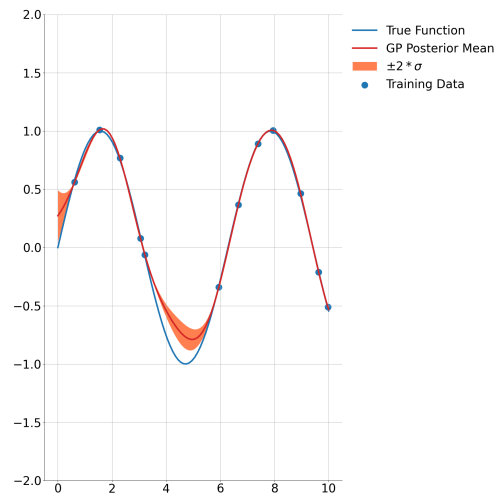
(b) Posterior Distribution over Functions for  $N_{tr} = 3$ (c) Posterior Distribution over Functions for  $N_{tr} = 12$ 

Figure 2.2: Given a true function to approximate,  $f = \sin(x)$ , Fig. 2.2a shows the prior distribution before any training data has been observed and Fig. 2.2b shows the posterior distribution once 3 noisy samples have been observed, Fig. 2.2c shows the posterior distribution once 12 noisy samples have been observed. Note the high variance along regions of the input space where no training data has been observed (*unexplored* regions) and the negligible variance in the neighborhood of the training points in Fig. 2.2b and 2.2c.

## Chapter 3

# Model Predictive Control with Gaussian Process Regression

In this chapter, the theoretical fundamentals of batch and real-time Model Predictive Control strategies combined with GP-learned functions are outlined with pertinent references.

### 3.1 Discrete-Time Model Predictive Control

*Model predictive control* (MPC) is an optimal control method for solving a *regulation problem* i.e., finding the best control input for a given dynamical system, objectives and constraints. The dynamic state model  $f$  is used to forecast system behavior over a given time horizon  $N$  and to optimize that predicted forecast over all possible horizon control inputs  $\mathbf{u}_0, \dots, \mathbf{u}_{N-1}$  [44].

The MPC horizon is divided between multiple *stages* corresponding to each time-step  $k_0 + k$ , where  $k_0$  is the absolute time-step and  $k = 0, 1, \dots, N - 1$  is the control horizon time-step. Each stage has an associated dynamic state equation, cost function, set of optimization variables and constraints. The final or *terminal* stage corresponds to a specific *terminal cost function* and *terminal set* for the terminal states.

The *feasible sets* for the stage state vectors  $\mathcal{X}_k$ , stage control input vectors  $\mathcal{U}_k$  and terminal state vector  $\mathcal{X}_f$  can be thought of as *implicit* constraints on the states and inputs at each time-step. In general, the constraints on the inputs are *hard constraints* in that the available system actuators are designed to operate within strict conditions and this is ensured by enforcing the inputs to remain within their given feasible sets. In contrast, the constraints on the states are often *soft constraints* in that it is preferable for the sake of stability and performance that the states remain within their given feasible sets, but it is not strictly required. The *inequality constraint functions*  $g_k$  represent time-varying *explicit* constraints on the system variables. The key difference between explicit and implicit constraints is that the former is converged towards with an associated dual variable in the gradient-based optimization method, while the latter is enforced by the projection operator.

The ideal MPC scheme has an *infinite horizon* i.e., the cost of a sequence of predicted states and control-inputs for each time-step in the infinite future is considered. It is clearly not computationally tractable to optimize over an infinite number of variables. The alternative approach employed is to define a terminal constraint set  $\mathcal{X}_f$  and a terminal cost function  $l_f : \mathbb{R}^{n_x} \mapsto \mathbb{R}$  which are imposed to approximate the effects of the constraints and cost over the remainder of the infinite control horizon.

Measuring disturbances introduces non-negligible delays and updating the control inputs requires actions that are implemented on digital control units. In *discrete-time* MPC (DTMPC), the temporal axis is discretized. This approach is applicable to systems in which the states are estimated and the control inputs applied at discrete time instances. If the MPC *sample time*  $\Delta t > 0$  is chosen to be sufficiently small, the behavior on time-scales shorter than this interval

can be safely ignored. A *difference equation*  $\mathbf{x}_{k+1} = f(\cdot)$  is used as the causal (i.e. dependent on past or current states, inputs and disturbances only) discrete-time dynamic state model. While an *auto-regressive* function can be modeled i.e. a difference equation that considers historic states, inputs and disturbances up to a given time-lag, in this work the simplified difference equation,  $\mathbf{x}_{k+1} = f(\tilde{\mathbf{x}}_k, \mathbf{u}_k; \mathbf{w}_k)$  is assumed. This relationship implies that the next state is a function of the current observed states  $\tilde{\mathbf{x}}_k$ , the current implemented control inputs  $\mathbf{u}_k$  and the current imposed exogenous inputs  $\mathbf{w}_k$ . The *sample number*  $k \in \mathcal{N}$  is a nonnegative integer which relates to the continuous time variable  $t$  as  $t = k\Delta t$ . The MPC optimal control problem is used to find the optimal set of control inputs over a full horizon such that the states and control inputs satisfy the implicit and explicit constraints, using a model of the system to predict its behavior. The optimal control input vector corresponding to the first time-step,  $\mathbf{u}_0^*$ , is then applied to the system, the outputs  $\mathbf{y}_k$  are measured and the state vector is estimated as  $\tilde{\mathbf{x}}_k$ . This process is illustrated in Fig. 3.1 and the equivalent optimal control problem is mathematically formulated in 3.1

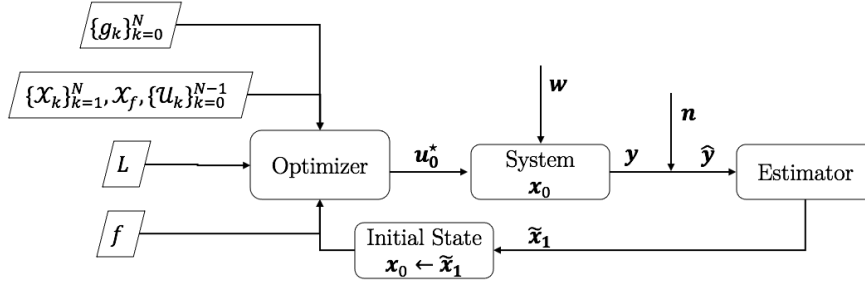


Figure 3.1: Model Predictive Control Block Diagram

$$\min_{\{\mathbf{x}_k\}_{k=1}^N, \{\mathbf{u}_k\}_{k=0}^{N-1}} L(\{\mathbf{x}_k\}_{k=1}^N, \{\mathbf{u}_k\}_{k=0}^{N-1}) := \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + l_f(\mathbf{x}_N) \quad (3.1a)$$

$$\text{s.t. } \mathbf{x}_0 = \mathbf{x}(t) \quad (3.1b)$$

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k; \mathbf{w}_k) \text{ for } k = 0, \dots, N-1 \quad (3.1c)$$

$$\mathbf{x}_k \in \mathcal{X}_k \text{ for } k = 0, \dots, N-1 \quad (3.1d)$$

$$\mathbf{u}_k \in \mathcal{U}_k \text{ for } k = 0, \dots, N-1 \quad (3.1e)$$

$$\mathbf{x}_N \in \mathcal{X}_f \quad (3.1f)$$

$$g_{k,j}(\mathbf{x}_k) \leq 0 \text{ for } k = 1, \dots, N; j = 1, \dots, n_g \quad (3.1g)$$

where:

$\mathbf{x}_k \in \mathbb{R}^{n_x}$  is the state vector at time-step  $k$

$\mathbf{u}_k \in \mathbb{R}^{n_u}$  is the (controllable) control input vector at time-step  $k$

$\mathbf{w}_k \in \mathbb{R}^{n_w}$  is the (uncontrollable) disturbance vector at time-step  $k$

$\mathbf{x}_0 \in \mathbb{R}^{n_x}$  is the given initial state vector

$\mathbf{x}(t)$  is the true initial state, as measured from the true system

$\mathbf{x}_N \in \mathbb{R}^{n_x}$  is the terminal state vector of the system at the final time-step of the horizon  $k = N$

$f: \mathbb{R}^{n_x+n_u+n_w} \mapsto \mathbb{R}^{n_x}$  is the discrete dynamic state function

$\mathcal{X}_k \subseteq \mathbb{R}^{n_x}$  is the time-varying implicit state feasible set at time-step  $k$

$\mathcal{U}_k \subseteq \mathbb{R}^{n_u}$  is the time-varying implicit control input feasible set at time-step  $k$

$\mathcal{X}_f \subseteq \mathbb{R}^{n_x}$  is the implicit terminal state feasible set

$L : \mathbb{R}^{N(n_x+n_u+n_w)} \mapsto \mathbb{R}$  is the *cost* (or objective) function to be minimized over all horizon state vectors  $\{\mathbf{x}_k\}_{k=1}^N$  and control input vectors  $\{\mathbf{u}_k\}_{k=0}^{N-1}$

$l : \mathbb{R}^{n_x+n_u+n_w} \mapsto \mathbb{R}$  is the time-invariant *stage cost* to be minimized over the stage state vector  $\mathbf{x}_k$  and input vector  $\mathbf{u}_k$  at each time step  $k = 0, \dots, N-1$  in the horizon

$l_f : \mathbb{R}^{n_x} \mapsto \mathbb{R}$  is the time-invariant *terminal cost* to be minimized over the terminal states  $x_N$

$g_{k,j} : \mathbb{R}^{n_x} \mapsto \mathbb{R}$  is the  $j^{\text{th}}$  time-varying explicit inequality constraint at time-step  $k$

The objective of a Model Predictive Controller is to design a control law [3.2](#):

$$\{\mathbf{u}_k\}_{k=0}^{N-1} = \kappa(\{\mathbf{x}_{k'}\}_{k'=0}^k) \quad (3.2)$$

such that the system described by the auto-regressive dynamic model [3.3](#)

$$\mathbf{x}_{k+1} = f(\{\mathbf{x}_{k'}\}_{k'=1}^k, \{\mathbf{u}_{k'}\}_{k'=0}^k; \mathbf{x}_0, \{\mathbf{w}_{k'}\}_{k'=0}^k) \quad (3.3)$$

and the control law [3.2](#) satisfy the following criteria:

1. The state vector  $\hat{\mathbf{x}}_k$  and the input vector  $\hat{\mathbf{u}}_k$  at each time-step  $k \in [0, N]$  satisfy their implicit and explicit constraints i.e:

$$\hat{\mathbf{x}}_k \in \mathcal{X}_k \forall k = 1, \dots, N \quad (3.4a)$$

$$\hat{\mathbf{x}}_N \in \mathcal{X}_f \quad (3.4b)$$

$$\hat{\mathbf{u}}_k \in \mathcal{U}_k \forall k = 0, \dots, N-1 \quad (3.4c)$$

$$g_{k,j}(\hat{\mathbf{x}}_k) \leq 0 \forall k = 1, \dots, N; j = 1, \dots, n_g \quad (3.4d)$$

In the receding horizon context of MPC, we wish to achieve *recursive feasibility* - the guaranteed existence of a feasible control input sequence at all time-steps  $k \geq 0$  when starting from a feasible initial state  $\hat{\mathbf{x}}_0$ .

2. The system is *stable* in the sense of Lyapunov i.e., if  $\bar{\mathbf{x}}$  is an *equilibrium point* such that  $f(\bar{\mathbf{x}}) = \bar{\mathbf{x}}$  and for every perturbation of  $\mathbf{x}_k$  from that equilibrium point for which the norm of the perturbation is upper-bounded by  $\epsilon > 0$ , there exists a scalar  $\delta$  such that if the normed distance of the initial state from the equilibrium point is bounded by  $\delta$ , the normed distance of all subsequent states from the equilibrium points is bounded by  $\epsilon$  i.e.:

$$\forall \epsilon > 0 \exists \delta > 0 \text{ s.t. if } \|\mathbf{x}_0 - \bar{\mathbf{x}}\| < \delta \rightarrow \|\mathbf{x}_k - \bar{\mathbf{x}}\| < \epsilon \forall k \geq 0 \quad (3.5)$$

Intuitively, this means that given that the initial state  $\mathbf{x}_0$  of the system is within a given radius  $\delta$  of the equilibrium point, the control law can eventually bring the state to within any desired distance  $\epsilon$  of the equilibrium point.

We wish to design a *stabilizing* control law such that [3.5](#) holds. However, a control law may be stabilizing but not robustly stabilizing. That is to say that arbitrary perturbations (or disturbances) could destabilize the system. A *robustly stable* controller is, in addition, stable for a given set of stochastic perturbations of the system.

3. The control law optimizes performance, as defined by a minimal cost function.
4. The set of initial states that can satisfy the above conditions,  $\{\mathbf{x}_0 \mid \text{Conditions 1 - 3}\}$  is maximized.

Uncertainty is introduced into the MPC problem when a system is time-varying in unpredictable ways, i.e. the cost function, constraints and/or system model change over time. *Adaptive* control is an approach to prevent performance and safety degradation resulting from model and objective uncertainty over time. However, adaptive control can introduce complex dynamics, requires a framework for updating the various components of the MPC and may require time-varying parameters. On the other hand, *non-adaptive* control assumes static MPC functions and parameterization, and is thus simpler to implement, but may be ineffective in a time-varying context.

Let  $\mathbf{z}_0 := \mathbf{u}_0$  be the *initial stage variables* for  $k = 0$ , let  $\mathbf{z}_k := [\mathbf{x}_k^T, \mathbf{u}_k^T]^T$  be the *stage variables* for  $k = 1, \dots, N - 1$  and let  $\mathbf{z}_N := \mathbf{x}_N$  be the *terminal variables* for  $k = N$  we optimize over. Note that the initial state vector  $\mathbf{x}_0$  and the disturbances  $\mathbf{w}_k$  at each time-step  $k$  are passed to the relevant functions as *parameters* because we take them as given and as such do not include them in the set of optimization variables. Let  $[g_{k,1}(\mathbf{x}_k), \dots, g_{k,n_g}(\mathbf{x}_k)]$  be the explicit inequality constraints associated with the states at time-step  $k$ . Additionally, the stage cost and terminal cost functions are assumed to be equivalent. Let  $\mathbf{y}_k$  be a vector of outputs at time-step  $k$  which is obtained from the function  $y(\mathbf{z}_k)$  and which is associated with a known stage cost function  $l^0(\mathbf{y}_k)$  and time-varying explicit inequality constraints  $[g_{k,1}^0(\mathbf{y}_k), \dots, g_{k,n_g}^0(\mathbf{y}_k)]$  for each time-step  $k$ . We assume that the dynamic state function  $f(\mathbf{z}_k; \mathbf{x}_0, \mathbf{w}_k)$  and a part of the stage cost  $l(\mathbf{z}_k)$  be unknown functions, which we replace with their posterior mean values,  $\bar{f}(\mathbf{z}_k; \mathbf{x}_0, \mathbf{w}_k)$  and  $\bar{l}(\mathbf{z}_k)$ , respectively. The MPC with Gaussian Processes can then be formulated as in [3.6](#).

$$\min_{\{\mathbf{z}_k\}_{k=0}^N} L(\{\mathbf{z}_k\}_{k=0}^N) := \sum_{k=0}^N \left[ \underbrace{\bar{l}(\mathbf{z}_k)}_{\text{unknown stage cost}} + \underbrace{l_k^0(\mathbf{y}_k)}_{\text{known stage cost}} \right] \quad (3.6a)$$

$$\text{s.t. } \mathbf{x}_0 = \mathbf{x}(t) \quad (3.6b)$$

$$\mathbf{x}_{k+1} = \underbrace{\bar{f}(\mathbf{z}_k; \mathbf{x}_0, \mathbf{w}_k)}_{\text{unknown dynamic state}} \quad \text{for } k = 0, \dots, N - 1 \quad (3.6c)$$

$$\mathbf{x}_k \in \mathcal{X}_k \text{ for } k = 0, \dots, N - 1 \quad (3.6d)$$

$$\mathbf{u}_k \in \mathcal{U}_k \text{ for } k = 0, \dots, N - 1 \quad (3.6e)$$

$$\mathbf{x}_N \in \mathcal{X}_f \quad (3.6f)$$

$$g_{k,j}(\mathbf{x}_k) \leq 0 \text{ for } k = 1, \dots, N; j = 1, \dots, n_g \quad (3.6g)$$

$$\mathbf{y}_k = y(\mathbf{z}_k; \mathbf{w}_k) \text{ for } k = 0, \dots, N \quad (3.6h)$$

$$g_{k,j}^0(\mathbf{y}_k) \leq 0 \text{ for } k = 1, \dots, N; j = 1, \dots, n_g^0 \quad (3.6i)$$

where:

$\bar{f} : \mathbb{R}^{n_x+n_u+n_w} \mapsto \mathbb{R}^{n_x}$  is the GP posterior mean of the unknown discrete-time dynamic state function

$\bar{l} : \mathbb{R}^{n_x+n_u+n_w} \mapsto \mathbb{R}$  is the the GP posterior mean of the unknown stage cost function

$l^0 : \mathbb{R}^{n_y} \mapsto \mathbb{R}$  is the known stage cost function

$\mathbf{y}_k \in \mathbb{R}^{n_y}$  is the vector of system outputs at time-step  $k$

$y : \mathbb{R}^{n_x+n_u+n_w} \mapsto \mathbb{R}^{n_y}$  is the known system output function

$g_{k,j} : \mathbb{R}^{n_x} \mapsto \mathbb{R}$  is the  $j^{\text{th}}$  time-varying inequality constraint function associated with the system states

$g_{k,j}^0 : \mathbb{R}^{n_y} \mapsto \mathbb{R}$  is the  $j^{\text{th}}$  time-varying inequality constraint function associated with the system outputs

## 3.2 Regularized Lagrangian Primal-Dual Gradient Optimization

We can optimally solve the MPC problem given in [3.6](#) with a *regularized primal-dual optimization method* implemented with a *batch* approach, or sub-optimally with an *online* (a.k.a *tracking*) approach (see [45](#) for a review of related time-varying optimization techniques). In this section, we will outline the theory behind this gradient-based optimization method and describe the algorithm. In the *batch* approach, the iterative algorithm is run for an arbitrarily large number of iterations until convergence is reached i.e. until the L2-norm of the difference between the optimization variables in sequential iterations falls below a given threshold, resulting in a *locally optimal* solution. In this section, we will outline the theory behind this optimization method and provide our reasons for employing it.

### 3.2.1 Strong Convexity

We define a strongly convex function as in Def. [3.2.1](#).

**Definition 3.2.1** (Strong Convexity, [46](#) Sec. 5.2). *A function  $f : \mathbb{E} \mapsto [-\infty, \infty)$  is called  $\sigma$ -strongly convex for a given  $\sigma > 0$  if  $\text{dom}(f)$  is convex and the following inequality holds for any  $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$  and  $\lambda \in [0, 1]$ :*

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}) - \frac{\sigma}{2} \lambda (1 - \lambda) \|\mathbf{x} - \mathbf{y}\|^2 \quad (3.7)$$

where:

$\mathbb{E}$  is the underlying Euclidean set, i.e. the L2-norm of  $\mathbf{x}$  is defined as  $\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$  for  $\mathbf{x} \in \mathbb{E}$

Def. [3.2.1](#) can be easily adapted to *strongly concave* functions by changing the signs of the terms in the inequality. We wish to have an objective function (the regularized Lagrangian function in our case) that is strongly convex in the primal variables (over which it is minimized) and strongly concave in the dual variables (over which it is maximized) in order to guarantee the existence and uniqueness of an optimizer, as stated by Theorem [3.2.2](#).

**Theorem 3.2.2** (Existence and Uniqueness of a Minimizer of Closed Strongly Convex Functions, [46](#) Theorem 5.25). *Let  $f : \mathbb{E} \mapsto (-\infty, \infty]$  be a proper, closed and  $\sigma$ -strongly convex function (for  $\sigma > 0$ ). Then:*

- a)  $f$  has a unique minimizer
- b)  $f(\mathbf{x}) - f(\mathbf{x}^*) \geq \frac{\sigma}{2} \|\mathbf{x} - \mathbf{x}^*\|^2$  for all  $\mathbf{x} \in \text{dom}(f)$ , where  $\mathbf{x}^*$  is the unique minimizer of  $f$ .

A convex function  $f$  is continuous at points in the interior of its domain,  $\text{int}(\text{dom}(f))$ , as stated by Theorem [3.2.3](#).

**Theorem 3.2.3** (Local Lipschitz Continuity of Convex Functions, [47](#) Theorem 7.36). *Let  $C \subseteq \mathbb{E}$  be a nonempty convex set. Let  $B[\mathbf{x}_0, \epsilon]$  be a norm ball centered at  $\mathbf{x}_0$  with radius  $\epsilon$ . Let  $f : \mathbb{E} \mapsto (-\infty, \infty]$  be a convex function. Let  $\mathbf{x}_0 \in \text{int}(\text{dom}(f))$ . Then there exist  $\epsilon > 0$  and  $L > 0$  such that  $B[\mathbf{x}_0, \epsilon] \subseteq C$  and:*

$$|f(\mathbf{x}) - f(\mathbf{x}_0)| \leq L \|\mathbf{x} - \mathbf{x}_0\| \quad (3.8)$$

for all  $\mathbf{x} \in B[\mathbf{x}_0, \epsilon]$ .

In general, convex functions are not continuous at their boundary points. However, the class of univariate closed and convex functions are continuous over their entire domain, as stated by Theorem [3.2.4](#).

**Theorem 3.2.4** (Continuity of Closed Convex Univariate Functions, [46] Theorem 2.22). *Let  $f : \mathbb{R} \mapsto (-\infty, \infty]$  be a proper closed and convex function. Then  $f$  is continuous over  $\text{dom}(f)$ .*

We can exploit Theorems [3.2.2] - [3.2.4] to apply a gradient-based optimization method to optimize a regularized Lagrangian function, as explained in Sec. [3.2.6]. We will first define the general gradient method in Sec. [3.2.2], the projected gradient method in Sec. [3.2.3] and the dual problem in Sec. [3.2.4].

### 3.2.2 The Gradient Method

Consider the unconstrained optimization problem [3.9]

$$\min f(\mathbf{x}) \mid \mathbf{x} \in \mathbb{E} \tag{3.9}$$

If  $f$  is differentiable over  $\mathbf{x}$ , a well-established method for solving this problem is the *gradient method*, described by Algorithm [1].

---

#### Algorithm 1 Gradient Method

---

**Input** Tolerance parameter,  $\nu > 0$

**Initialization** Pick initial state  $\mathbf{x}_0 \in \mathbb{R}^n$  arbitrarily

**General Step**

```

for  $\tau = 1, 2, \dots$  do
  Pick a step-size  $\alpha_\tau > 0$ 
  Set  $\mathbf{x}_{\tau+1} \leftarrow \mathbf{x}_\tau - \alpha_\tau \nabla f(\mathbf{x}_\tau)$ 
  if  $\|\nabla f(\mathbf{x}_{\tau+1})\| \leq \nu$  then
    return  $\mathbf{x}_{\tau+1}$ 

```

---

The negative of the function gradient  $-\nabla f(\mathbf{x}_\tau)$  is called the *descent direction* and is defined in Def. [3.2.5].

**Definition 3.2.5** (Descent Direction, [46] Sec. 8.1.1). *Let  $f : \mathbb{E} \mapsto (-\infty, \infty]$  be an extended real-valued function. Let  $\mathbf{x} \in \text{int}(\text{dom}(f))$ . A vector  $\mathbf{0} \neq \mathbf{d} \in \mathbb{E}$  is called a descent direction of  $f$  if the directional derivative  $f'(\mathbf{x}; \mathbf{d})$  exists and is negative.*

A key property of a descent direction is that updating the optimization variables  $\mathbf{x}$  along this direction in sufficiently small steps will lead to a decrease in the function value  $f(\mathbf{x})$ , as stated by Lemma [3.2.6].

**Lemma 3.2.6** (Descent Property of Descent Directions, [47] Lemma 4.2). *Let  $f : \mathbb{E} \mapsto (-\infty, \infty]$  be an extended real-valued function. Let  $\mathbf{x} \in \text{int}(\text{dom}(f))$ , and assume that  $\mathbf{0} \neq \mathbf{d} \subseteq \mathbb{E}$  is a descent direction of  $f$  at  $\mathbf{x}$ . Then there exists  $\epsilon > 0$  such that  $\mathbf{x} + \alpha \mathbf{d} \in \text{dom}(f)$  and*

$$f(\mathbf{x} + \alpha \mathbf{d}) < f(\mathbf{x}) \tag{3.10}$$

for any  $\alpha \in (0, \epsilon]$ .

### 3.2.3 The Projected Gradient Method

An extension of the gradient method to solve constrained optimization problems is the *projected gradient method*, described by Algorithm [2]. In this case, each iteration  $\tau$  consists of a step taken along the negative direction of the gradient  $\nabla f(\mathbf{x}_\tau)$  followed by an orthogonal projection onto the underlying set  $C$  (see [3.11]). The projection ensures that subsequent iterates  $\mathbf{x}_{\tau+1}$  are contained by their feasible set  $C$ .

$$\mathcal{P}_C(\mathbf{x}) = \arg \min_{\mathbf{y} \in C} \|\mathbf{y} - \mathbf{x}\| \quad (3.11)$$

---

**Algorithm 2** Projected Gradient Method
 

---

**Input** Tolerance parameter,  $\nu > 0$

**Initialization** Pick initial state  $\mathbf{x}_0 \in C$  arbitrarily

**General Step**

**for**  $\tau = 1, 2, \dots$  **do**  
 Pick a step-size  $\alpha_\tau > 0$   
 $\mathbf{x}_{\tau+1} \leftarrow P_C(\mathbf{x}_\tau - \alpha_\tau \nabla f(\mathbf{x}_\tau))$   
**if**  $\|\nabla f(\mathbf{x}_{\tau+1})\| \leq \nu$  **then**  
   **return**  $\mathbf{x}_{\tau+1}$

---

### 3.2.4 The Dual Problem

This section draws from [48] Chapter 5. Consider the constrained optimization problem in [3.12]

$$\min_{\mathbf{x} \in \mathcal{D}} f_0(\mathbf{x}) \quad (3.12a)$$

$$\text{s.t. } f_i(\mathbf{x}) \leq 0 \forall i = 1, \dots, m \quad (3.12b)$$

$$h_i(\mathbf{x}) = 0 \forall i = 1, \dots, p \quad (3.12c)$$

where:

$\mathbf{x} \in \mathbb{R}^n$  are the optimization variables

$f(\mathbf{x}^*)$  is the optimal value

$\mathcal{D} = \bigcap_{i=0}^m \text{dom}(f_i) \cap \bigcap_{i=1}^p \text{dom}(h_i)$  is the nonempty domain of feasible values of  $\mathbf{x}$

We can implicitly take the constraints [3.12b] - [3.12c] into account by defining the *Lagrangian function* associated with problem [3.12], which is the original objective function augmented with a weighted sum of the constraint functions, given by [3.13]

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f_0(\mathbf{x}) + \boldsymbol{\lambda}^T \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} + \boldsymbol{\mu}^T \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_p(\mathbf{x}) \end{bmatrix} \quad (3.13)$$

where:

$\boldsymbol{\lambda} \in \mathbb{R}^m$  are the *dual variables* or *Lagrangian multipliers* associated with the inequality constraints  $f_1(\mathbf{x}), \dots, f_m(\mathbf{x})$

$\boldsymbol{\mu} \in \mathbb{R}^p$  are the dual variables or Lagrangian multipliers associated with the equality constraints  $h_1(\mathbf{x}), \dots, h_p(\mathbf{x})$

The *Lagrangian dual function* (or simply the *dual function*) is defined as the minimum value of the Lagrangian over the decision variables  $\mathbf{x}$ , given by [3.14]



$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \inf_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x} \in \mathcal{D}} \left( f_0(\mathbf{x}) + \boldsymbol{\lambda}^T \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} + \boldsymbol{\mu}^T \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_p(\mathbf{x}) \end{bmatrix} \right) \quad (3.14)$$

Notice that since [3.14](#) is the pointwise infimum (over  $\mathbf{x}$ ) of a family of affine functions of the dual variables  $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ , it is a concave function (see Appendix [C.1](#)), even if the primal problem [3.12](#) is non-convex (see Appendix [C.2](#)).

Assuming that  $\boldsymbol{\lambda} \succeq \mathbf{0}$ , it can be shown that the dual function yields a lower bound on the optimal value of the primal problem. Let  $\tilde{\mathbf{x}}$  be a feasible point for the problem [3.12](#), such that  $f_i(\tilde{\mathbf{x}}) \leq 0 \forall i = 1, \dots, m$  and  $h_i(\tilde{\mathbf{x}}) = 0 \forall i = 1, \dots, p$ . Then:

$$\underbrace{\boldsymbol{\lambda}^T \begin{bmatrix} f_1(\tilde{\mathbf{x}}) \\ \vdots \\ f_m(\tilde{\mathbf{x}}) \end{bmatrix}}_{\leq 0} + \underbrace{\boldsymbol{\mu}^T \begin{bmatrix} h_1(\tilde{\mathbf{x}}) \\ \vdots \\ h_p(\tilde{\mathbf{x}}) \end{bmatrix}}_{=0} \leq 0 \quad (3.15)$$

If we add the feasible objective function  $f_0(\tilde{\mathbf{x}})$  to each side we have an inequality constraint on the Lagrangian function:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f_0(\tilde{\mathbf{x}}) + \boldsymbol{\lambda}^T \begin{bmatrix} f_1(\tilde{\mathbf{x}}) \\ \vdots \\ f_m(\tilde{\mathbf{x}}) \end{bmatrix} + \boldsymbol{\mu}^T \begin{bmatrix} h_1(\tilde{\mathbf{x}}) \\ \vdots \\ h_p(\tilde{\mathbf{x}}) \end{bmatrix} \leq f_0(\tilde{\mathbf{x}}) \\ &\Rightarrow \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f_0(\tilde{\mathbf{x}}) \end{aligned} \quad (3.16)$$

The dual function is the lower bound of the Lagrangian over all primal variables by definition, so we have:

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) &= \inf_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq \mathcal{L}(\tilde{\mathbf{x}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f_0(\tilde{\mathbf{x}}) \\ &\Rightarrow g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f_0(\tilde{\mathbf{x}}) \\ &\Rightarrow g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f_0(\mathbf{x}^*) \end{aligned} \quad (3.17)$$

Note the significance of the positive definite constraint on the inequality dual variables. In order for the dual function [3.14](#) to give a finite ( $> -\infty$ ) lower bound on the original problem, we must have *dual feasible* Lagrangian multipliers i.e.  $\boldsymbol{\lambda} \succeq \mathbf{0}$  and  $\boldsymbol{\lambda}, \boldsymbol{\mu} \in \text{dom}(g)$ .

The objective of the *dual problem* is to find the worst-case lower bound on the original objective function  $f_0(\mathbf{x})$ , or in other words, to maximize the dual function over the dual feasible Lagrangian multipliers. This is formulated by the *Lagrangian dual problem* in [3.18](#).

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\mu} \in \text{dom}(g)} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \quad (3.18a)$$

$$\text{s.t. } \boldsymbol{\lambda} \succeq \mathbf{0} \quad (3.18b)$$

The solution of [3.18](#),  $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ , are the *dual optimal* variables. The property [3.17](#) is known as *weak duality* and the (nonnegative) difference  $f_0(\mathbf{x}^*) - g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$  is known as the *duality gap*. If the duality gap is zero i.e.  $f_0(\mathbf{x}^*) = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ , then we have the property of *strong duality*. Strong duality does not, in general, hold, but under certain conditions it can be show to. One such qualification is *Slater's condition*, which requires that the problem be convex (see Appendix [C.2](#)) and  $\mathbf{x} \in \text{relint}(\mathcal{D})$  (see [48](#) Sec. 5.2.3). The significance of strong duality is that the optimal cost of both the primal problem [3.12](#) and the dual problem [3.18](#) are equivalent.

Consider now the MPC problem given by [3.6](#). Let  $\mathcal{Z}$  be the feasible set for the horizon optimization variables  $\{\mathbf{z}_k\}_{k=0}^N$ . Let  $\mathbb{R}_+^{N(n_g+n_g^0)}$  be the nonnegative orthant in  $\mathbb{R}^{N(n_g+n_g^0)}$  and the feasible set for the dual inequality variables  $\boldsymbol{\lambda}$  (corresponding to the constraints at each time-step in the horizon for the system states and for the system outputs). Let the dual equality variables  $\boldsymbol{\mu}$  be unconstrained with a feasible set of  $\mathbb{R}^{Nn_x}$  (corresponding the dynamic state difference equation for each state at each time-step in the horizon). A *feasible solution* to the problem is given by the *Karush-Kuhn-Tucker (KKT) point*  $\tilde{\mathbf{q}} := (\tilde{\mathbf{z}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}) \in \mathcal{Z} \times \mathbb{R}_+^{N(n_g+n_g^0)} \times \mathbb{R}^{Nn_x}$  (see [Appendix C.3](#) for background information on the conditions that must be satisfied by a KKT point). We define the *Lagrangian function* for the problem as in [Eqn. 3.19](#).

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := L(\mathbf{z}) + \boldsymbol{\lambda}^T G(\mathbf{z}) + \boldsymbol{\mu}^T F(\mathbf{z}) \quad (3.19)$$

where:

$\mathbf{z} \in \mathbb{R}^{N(n_x+n_u)}$  is the vector of optimization variables  $[\mathbf{u}_0^T, \mathbf{x}_1^T, \mathbf{u}_1^T, \mathbf{x}_2^T, \dots, \mathbf{u}_{N-1}^T, \mathbf{x}_N^T]^T$ , also known as the *primal variables*

$L : \mathbb{R}^{N(n_x+n_u)} \mapsto \mathbb{R}$  is the horizon cost function given by [3.6a](#)

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_{1,1}(\mathbf{x}_1) \\ \vdots \\ g_{1,n_g}(\mathbf{x}_1) \\ g_{2,1}(\mathbf{x}_2) \\ \vdots \\ g_{2,n_g}(\mathbf{x}_2) \\ \vdots \\ g_{N,n_g}(\mathbf{x}_N) \end{bmatrix} \leq \mathbf{0} \text{ is the vector of time-varying explicit state inequality constraints}$$

$$\mathbf{g}^0(\mathbf{y}) = \begin{bmatrix} g_{1,1}^0(\mathbf{x}_1) \\ \vdots \\ g_{1,n_g^0}^0(\mathbf{x}_1) \\ g_{2,1}^0(\mathbf{x}_2) \\ \vdots \\ g_{2,n_g^0}^0(\mathbf{x}_2) \\ \vdots \\ g_{N,n_g^0}^0(\mathbf{x}_N) \end{bmatrix} \leq \mathbf{0} \text{ is the vector of time-varying explicit system output inequality constraints}$$

$$G(\mathbf{z}) = \begin{bmatrix} \mathbf{g}(\mathbf{x}) \\ \mathbf{g}^0(\mathbf{y}) \end{bmatrix} \text{ is the set of all explicit inequality constraint functions}$$

$\boldsymbol{\lambda} \in \mathbb{R}^{N(n_g+n_g^0)}$  is the vector of *inequality dual variables* associated with each explicit inequality constraint contained in  $G$

$$F(\mathbf{z}) = \begin{bmatrix} \mathbf{x}_1 - \bar{f}(\mathbf{z}_0) \\ \vdots \\ \mathbf{x}_N - \bar{f}(\mathbf{z}_{N-1}) \end{bmatrix} = \mathbf{0} \text{ is the set of explicit dynamic state equality constraint functions}$$

$\boldsymbol{\mu} \in \mathbb{R}^{Nn_x}$  is the vector of *equality dual variables* associated with each equality constraint contained in  $F$

### 3.2.5 Regularized Lagrangian Function

The Lagrangian function is *regularized* by adding a term that renders the function strongly concave in the dual variable vectors, see [3.20](#). This strongly concave regularization is necessary to derive bounded Q-linear convergence for the tracking error of the real-time gradient-based optimization algorithm [12](#).

$$\mathcal{L}^T(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := L(\mathbf{z}) + \boldsymbol{\lambda}^T G(\mathbf{z}) + \boldsymbol{\mu}^T F(\mathbf{z}) - \underbrace{\frac{\epsilon}{2} (\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{\text{prior}}\| + \|\boldsymbol{\mu} - \boldsymbol{\mu}_{\text{prior}}\|)}_{\text{regularization term}} \quad (3.20)$$

The effect of the regularization term is firstly to ensure a unique solution to the dual problem and secondly to drive the dual variables  $\boldsymbol{\lambda}$  and  $\boldsymbol{\mu}$  to their prior estimates  $\boldsymbol{\lambda}_{\text{prior}}$  and  $\boldsymbol{\mu}_{\text{prior}}$ , respectively.

### 3.2.6 Primal-Dual Gradient Method

In this section, we will first outline the mathematical prerequisites to understanding the primal-dual gradient method employed in this work and then proceed to describe the algorithm itself.

For a twice continuously differentiable real-valued function  $f$ , the *Hessian* of  $f(\mathbf{z})$  at  $\hat{\mathbf{z}}$  is denoted by  $\nabla^2 f(\hat{\mathbf{z}})$ . For a function  $f(\mathbf{z})$  that is continuously differentiable in  $\mathbf{z}$ , its *gradient* with respect to  $\mathbf{z}$  at  $\hat{\mathbf{z}}$  is denoted by  $\nabla_{\mathbf{z}}^2 f(\hat{\mathbf{z}})$ .

For a continuously differentiable vector-valued function  $f(\mathbf{z})$ , its *Jacobian* matrix evaluated at  $\hat{\mathbf{z}}$  is denoted by  $J_f(\hat{\mathbf{z}})$ . Given a convex set  $C$ , its *normal cone* at  $\mathbf{z} \in C$ , defined by  $\{y : y^T(\mathbf{c} - \mathbf{z}) \leq 0, \forall \mathbf{c} \in C\}$ , is denoted by  $N_C(\mathbf{z})$ . It has been established that the projection operator [3.11](#) exists and is unique when the underlying set  $C$  is nonempty, closed and convex [49](#).

$\mathbb{R}_+ = [0, +\infty)$  and  $\mathbb{R}_{++} = (0, +\infty)$  denote nonnegative and positive real numbers, respectively.  $I$  denotes the identity matrix, or  $I_n \in \mathbb{R}^{n \times n}$  when the dimensions need to be specified. The L2-norm of  $\mathbf{z}$  is denoted by  $\|\mathbf{z}\| := \sqrt{\mathbf{z}^T \mathbf{z}}$ .

We make the following assumptions regarding the optimal control problem [3.6](#):

**Assumption 3.2.7.** For the problem [3.6](#), it holds that:

[3.2.7a](#))  $\mathcal{Z} := \mathbb{R}^{N(n_x+n_u)}$  is convex and closed.

[3.2.7b](#)) The functions  $L$  and  $G$  are twice continuously differentiable over  $\mathbf{z} \in \mathcal{Z}$ . Additionally,  $\nabla_{\mathbf{z}}^2 L(\mathbf{z})$  and  $\nabla_{\mathbf{z}}^2 G(\mathbf{z})$  are continuous over  $\mathbf{z} \in \mathcal{Z}$ .

[3.2.7c](#)) There exists a Lipschitz continuous trajectory  $\mathbf{q}^* = (\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$  such that:

[3.2.7c](#))a)  $\mathbf{q}^* \in \mathcal{Z} \times \mathbb{R}_+^{N(n_g+n_g^0)} \times \mathbb{R}^{Nn_x}$

[3.2.7c](#))b)  $\nabla_{\mathbf{z}} L(\mathbf{z}^*) + J_{G,\mathbf{z}}(\mathbf{z}^*)^T \boldsymbol{\lambda}^* + J_{F,\mathbf{z}}(\mathbf{z}^*)^T \boldsymbol{\mu}^* \in -N_{\mathcal{Z}}(\mathbf{z}^*)$

[3.2.7c](#))c)  $G(\mathbf{z}^*) \in N_{\mathbb{R}_+^{N(n_g+n_g^0)}}(\boldsymbol{\lambda}^*)$

[3.2.7c](#))d)  $F(\mathbf{z}^*) \in \mathbf{0}$

An update is performed at each iteration  $\tau$  until convergence to a pre-defined tolerance  $\nu$  or a maximum number of iterations `MaxIter` is reached.

Let  $\hat{\mathbf{q}}_0 = (\hat{\mathbf{z}}_0, \hat{\boldsymbol{\lambda}}_0, \hat{\boldsymbol{\mu}}_0) \in \mathcal{Z} \times \mathbb{R}_+^{N(n_g+n_g^0)} \times \mathbb{R}^{Nn_x}$  be a feasible initial KKT point. The regularized primal-dual gradient algorithm iteratively updates the vector  $\hat{\mathbf{q}}_\tau$  as in [3.21](#). The algorithm is illustrated in Fig. [3.2](#).

$$\hat{\mathbf{z}}_\tau = \mathcal{P}_{\mathcal{Z}} \left[ \hat{\mathbf{z}}_{\tau-1} - \alpha \left( \nabla L(\hat{\mathbf{z}}_{\tau-1}) + J_G(\hat{\mathbf{z}}_{\tau-1})^T \hat{\boldsymbol{\lambda}}_{\tau-1} + J_F(\hat{\mathbf{z}}_{\tau-1})^T \hat{\boldsymbol{\mu}}_{\tau-1} \right) \right] \quad (3.21a)$$

$$\hat{\boldsymbol{\lambda}}_\tau = \mathcal{P}_{\mathbb{R}_+^{N(n_g+n_g^0)}} \left[ \hat{\boldsymbol{\lambda}}_{\tau-1} + \eta \alpha \left( G(\hat{\mathbf{z}}_\tau) - \epsilon (\hat{\boldsymbol{\lambda}}_{\tau-1} - \boldsymbol{\lambda}_{\text{prior}}) \right) \right] \quad (3.21b)$$

$$\hat{\boldsymbol{\mu}}_\tau = \hat{\boldsymbol{\mu}}_{\tau-1} + \eta \alpha \left( F(\hat{\mathbf{z}}_\tau) - \epsilon (\hat{\boldsymbol{\mu}}_{\tau-1} - \boldsymbol{\mu}_{\text{prior}}) \right) \quad (3.21c)$$

where:

$\hat{\mathbf{z}}_\tau$  is the primal variable solution at the  $\tau^{\text{th}}$  iteration

$\alpha$  is the step-size for the primal variable updates

$\eta\alpha$  is the step-size for the dual variable update

$\epsilon$  is a parameter which controls the amount of regularization on the dual variables

$\mathcal{Z} := \mathcal{U}_0 \times \mathcal{X}_1 \times \dots \times \mathcal{U}_{N-1} \times \mathcal{X}_f$  is the feasible set of primal variables

The step-sizes  $\alpha$  and  $\eta$  could be given by vectors, where each element corresponds to a step-size specific to that dimension of  $\mathbf{z}$ ,  $\boldsymbol{\lambda}$  and  $\boldsymbol{\mu}$ , respectively. In this work, they are left as scalars for simplicity.

Note that a projected gradient update is applied to the constrained primal variables  $\mathbf{z}$  and to the constrained inequality dual variables  $\boldsymbol{\lambda}$ , while a non-projected gradient update is applied to the unconstrained equality dual variables  $\boldsymbol{\mu}$ .

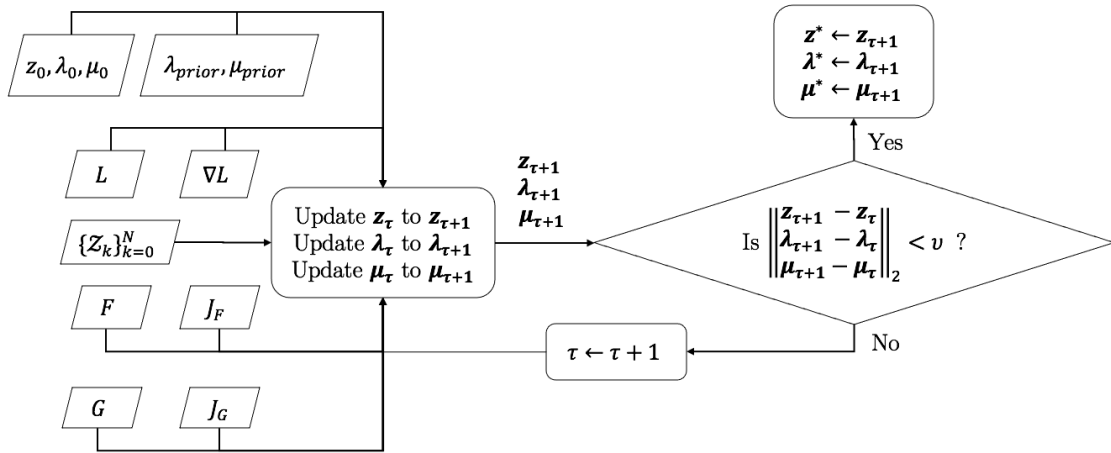


Figure 3.2: Primal-Dual Batch Optimization Flowchart

Consider a system of  $D$  controllable devices, where the dynamic state equation and the stage cost corresponding to each device are learned using GPs (this is an example of *dual control* [50], in which the system is simultaneously being identified and controlled). The algorithm for integrating the learned functions with the described primal-dual update is given by [3.21].

Before the controller is implemented, each GP (one for each unknown state and each cost function of each device  $d$  in the network), is *initialized* with a pre-defined number of training samples  $N_{tr,0}$ . The resulting initial training sets  $((\mathbf{Z}^d, \mathbf{g}_i^d)$  for each state  $i$  of each device  $d$  and  $(\mathbf{Z}^d, \mathbf{j}^d)$  for each cost function of each device  $d$ ) are subsequently passed to the controller.

The control system is simulated for  $T$  time-steps. At the beginning of each time-step  $k_0$ , if the training sampling time has passed for the state ( $\Delta t_x^d$ ) or cost ( $\Delta t_l^d$ ) associated with a device, then the number (depending on how many multiples of the sampling time have passed) of training points collected from the *real-time* system/user feedback is added to the *online* subset of the appropriate training dataset, which is then truncated to only include the most recent  $N_{tr} - N_{tr,0}$  data points. The *offline* subset of the  $N_{tr,0}$  training data set is left untouched. With this approach, we can perform an initial offline training of the functions with  $N_{tr,0}$  training inputs selected to capture as much information as possible and retain this input data while we incrementally update  $N_{tr} - N_{tr,0}$  additional training samples during operation based on live feedback. The result is that the offline subset can *explore* the input space by capturing a large range of the variability of the function (if the training samples are distributed carefully) while the online subset can fine-tune

the training set based on short-term trends by learning more about the input space it is moving towards and through in real-time from feedback. The hyperparameters  $\theta$  of the covariance kernel are optimized for by maximizing the marginal log likelihood and the  $K(X, X) + \sigma_n^2 I$  matrix is inverted to reflect the new training data. If new device state feedback is available, the device feasible sets  $\mathcal{X}^d$  and  $\mathcal{U}^d$  are also updated to partially exclude the unexplored, and thus unsafe, regions of the updated training set. The primal-dual method is executed until the iterates have converged or the maximum number of iterations has been reached. The resulting KKT point  $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$  is then time-shifted by removing the variables corresponding to the first horizon stage and repeating the variables corresponding to the terminal stage.

The full MPC with Gaussian Processes and the primal-dual optimization algorithm is outlined in Algorithm 3. In the batch case, the convergence is terminated once the L2-norm of the difference between consecutive iterates has fallen below a pre-defined threshold,  $\nu$ . In the *online* implementation of this algorithm, described in the next section, the convergence is terminated once a pre-defined number of iterations, `MaxIter` has been reached.

### 3.3 Online Regularized Lagrangian Primal-Dual Gradient Optimization

Perfect tracking could be achieved by solving the problem 3.6 to convergence at each time-step (as in the batch approach described in Sec. 3.2). However, if a) the objectives, constraints, dynamic state and/or output functions are *time-varying* and we need to set the MPC sampling time  $\Delta t$  to a scale faster than these variations, or b) an optimized control input is required at a high sampling rate, there may not be sufficient time available for the algorithm to converge. The optimization procedure outlined in Sec. 3.2 would not be practical in these cases. Additionally, suppose there is limited (or burdensome) observability of the disturbances  $\mathbf{w}_k$  on which the outputs  $\mathbf{y}_k$  depend. In this scenario, we would not be able to compute either the outputs using the closed-form expression nor their gradients. If the gradient-based algorithm performs only a single iteration however, we could employ *measurements* of the outputs  $\hat{\mathbf{y}}$  as placeholders for their synthetic functions  $y(\mathbf{z}_k; \mathbf{w}_k)$  and employ *estimated gradients* based on these measurements instead of computing the true gradients in the gradient-based method. *Real-time* MPC (illustrated in Fig. 3.3) can be implemented with an *online* optimization algorithm, and is an alternative approach which aims to solve the problem in as few iterations as possible. It is inherently *sub-optimal*, in that the optimization algorithm is terminated once a given (typically  $\leq 100$ ) number of iterations have been completed, as opposed to terminating the algorithm once the norm of the change in the iterates has fallen below a given threshold. It is important then, to consider how the algorithm can be adapted to converge towards the true optimum in as few iterations as possible for each time-step. In this section, we will describe core concepts of the online implementation of Algorithm 3 proposed in this work.

#### 3.3.1 Tracking Error

Let the horizon cost function  $L$ , the explicit inequality constraints  $G$  and the explicit equality constraints  $F$  be *time-varying*, such that at each instance or absolute point in time at which 3.6 is run,  $k_0 = 0, 1, \dots, T - 1$ , the functions  $L$ ,  $G$  and  $F$  depend on this absolute time-step, see 3.22

$$L(\mathbf{z}) := L_{k_0}(\mathbf{z}_{k_0}) \tag{3.22a}$$

$$G(\mathbf{z}) := G_{k_0}(\mathbf{z}_{k_0}) \tag{3.22b}$$

$$F(\mathbf{z}) := F_{k_0}(\mathbf{z}_{k_0}) \tag{3.22c}$$

where:

$\mathbf{z}_{k_0}$  is the vector of optimization variables at the  $k_0^{\text{th}}$  instance of the MPC run

**Algorithm 3** Primal-Dual MPC Algorithm with Gaussian Processes

---

```

1: procedure OPTIMIZE(
   $T, \text{MaxIter}, \nu, \alpha, \eta, \epsilon, \boldsymbol{\lambda}_{\text{prior}}, \boldsymbol{\mu}_{\text{prior}}, L, F, G, \mathcal{Z}, \mathbf{x}_0$ ;
   $(Z_{\text{offline}}^d, \mathbf{g}_i^d), \sigma_{g_i^d, n}^2$  for each state  $i$  and each device  $d$ ;  $(Z_{\text{offline}}^d, \mathbf{j}^d), \sigma_{j_k^d, n}^2$  for each device  $d$  )
2:    $\mathbf{z}_0 \leftarrow \mathbf{x}_0$  for states,  $\mathbf{0}$  for control inputs ▷ initialize primal variables
3:    $\boldsymbol{\lambda}_0 \leftarrow [\mathbf{0}_{N(n_g+n_g^0)}]$  ▷ initialize dual inequality variables
4:    $\boldsymbol{\mu}_0 \leftarrow [\mathbf{0}_{Nn_x}]$  ▷ initialize dual equality variables
5:   for each time-step  $k_0 = 0, 1, \dots, T-1$  do
6:     for  $d = 1, \dots, D$  do ▷ for each device/user
7:       if device  $d$  state feedback received then
8:          $Z_{\text{online}}^d \leftarrow$  last  $N_{tr} - N_{tr,0}$  rows of  $\begin{bmatrix} Z_{\text{online}}^d \\ \mathbf{z}_k^d \end{bmatrix}$  ▷ update device  $d$  online input
          training set
9:          $Z^d \leftarrow \begin{bmatrix} Z_{\text{offline}}^d \\ Z_{\text{online}}^d \end{bmatrix}$  ▷ update device  $d$  full input training set
10:        for  $i = 1, \dots, n_x^d$  do ▷ for each device state
11:           $\mathbf{g}_{i,\text{online}}^d \leftarrow$  last  $N_{tr} - N_{tr,0}$  rows of  $\begin{bmatrix} \mathbf{g}_{i,\text{online}}^d \\ g_k^d \end{bmatrix}$  ▷ update device  $d$  state  $i$ 
            online output training set
12:           $\mathbf{g}_i^d \leftarrow \begin{bmatrix} \mathbf{g}_{i,\text{offline}}^d \\ \mathbf{g}_{i,\text{online}}^d \end{bmatrix}$  ▷ update device  $d$  state  $i$  full output training set
13:           $\boldsymbol{\theta} \leftarrow \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{g}_i^d | Z^d, \boldsymbol{\theta})$  ▷ optimize for hyperparameters
14:          invert  $K(Z^d, Z^d) + \sigma_{g_i^d, n}^2 I$ 
15:           $Z^d \leftarrow Z_{\text{orig}}^d \cap Z^d$  ▷ update device state/input feasible sets based on input
            training set
16:          if user  $d$  cost feedback received then
17:             $Z_{\text{online}}^d \leftarrow$  last  $N_{tr} - N_{tr,0}$  rows of  $\begin{bmatrix} Z_{\text{online}}^d \\ \mathbf{z}_k^d \end{bmatrix}$  ▷ update user  $d$  cost online input
              training set
18:             $Z^d \leftarrow \begin{bmatrix} Z_{\text{offline}}^d \\ Z_{\text{online}}^d \end{bmatrix}$  ▷ update user  $d$  cost full input training set
19:             $\mathbf{j}_{\text{online}}^d \leftarrow$  last  $N_{tr} - N_{tr,0}$  rows of  $\begin{bmatrix} \mathbf{j}_{\text{online}}^d \\ j_k^d \end{bmatrix}$  ▷ update user  $d$  cost online output
              training set
20:             $\mathbf{j}^d \leftarrow \begin{bmatrix} \mathbf{j}_{\text{offline}}^d \\ \mathbf{j}_{\text{online}}^d \end{bmatrix}$  ▷ update user  $d$  cost full output training set
21:             $\boldsymbol{\theta} \leftarrow \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{j}^d | Z^d, \boldsymbol{\theta})$  ▷ optimize for hyperparameters
22:            invert  $K(Z^d, Z^d) + \sigma_{j_k^d, n}^2 I$ 
23:           $\tau \leftarrow 0$ 
24:          while  $\tau < \text{MaxIter}$  do
25:             $\mathbf{z}_{\tau+1} \leftarrow \mathcal{P}_{\mathcal{Z}} [\mathbf{z}_{\tau} - \alpha (\nabla L(\mathbf{z}_{\tau}) + J_G(\mathbf{z}_{\tau})^T \boldsymbol{\lambda}_{\tau} + J_F(\mathbf{z}_{\tau})^T \boldsymbol{\mu}_{\tau})]$  ▷ update primal
              variables
26:             $\boldsymbol{\lambda}_{\tau+1} \leftarrow \mathcal{P}_{\mathbb{R}_+^{n_g+n_g^0}} [\boldsymbol{\lambda}_{\tau} + \eta \alpha (G(\mathbf{z}_{\tau+1}) - \epsilon(\boldsymbol{\lambda}_{\tau} - \boldsymbol{\lambda}_{\text{prior}}))]$  ▷ update dual inequality
              variables
27:             $\boldsymbol{\mu}_{\tau+1} \leftarrow \boldsymbol{\mu}_{\tau} + \eta \alpha (F(\mathbf{z}_{\tau+1}) - \epsilon(\boldsymbol{\mu}_{\tau} - \boldsymbol{\mu}_{\text{prior}}))$  ▷ update dual equality variables
28:            if  $\left\| \begin{bmatrix} \mathbf{z}_{\tau+1} - \mathbf{z}_{\tau} \\ \boldsymbol{\lambda}_{\tau+1} - \boldsymbol{\lambda}_{\tau} \\ \boldsymbol{\mu}_{\tau+1} - \boldsymbol{\mu}_{\tau} \end{bmatrix} \right\|_2 < \nu$  then
29:               $\mathbf{z}^* \leftarrow \mathbf{z}_{\tau+1}$  ▷ assign optimal primal variables
30:               $\boldsymbol{\lambda}^* \leftarrow \boldsymbol{\lambda}_{\tau+1}$  ▷ assign optimal dual inequality variables
31:               $\boldsymbol{\mu}^* \leftarrow \boldsymbol{\mu}_{\tau+1}$  ▷ assign optimal dual equality variables
32:              break
33:             $\tau \leftarrow \tau + 1$  ▷ increment iteration count
34:             $\mathbf{x}_{k+1} \leftarrow f_{\text{true}}(\mathbf{u}_0^*)$  ▷ input optimal first control input into true system to get next state
35:             $\mathbf{z}_0 \leftarrow$  time-shifted  $\mathbf{z}_k^*$  ▷ warm-start primal variables
36:             $\boldsymbol{\lambda}_0 \leftarrow$  time-shifted  $\boldsymbol{\lambda}_k^*$  ▷ warm-start dual inequality variables
37:             $\boldsymbol{\mu}_0 \leftarrow$  time-shifted  $\boldsymbol{\mu}_k^*$  ▷ warm-start dual equality variables

```

---

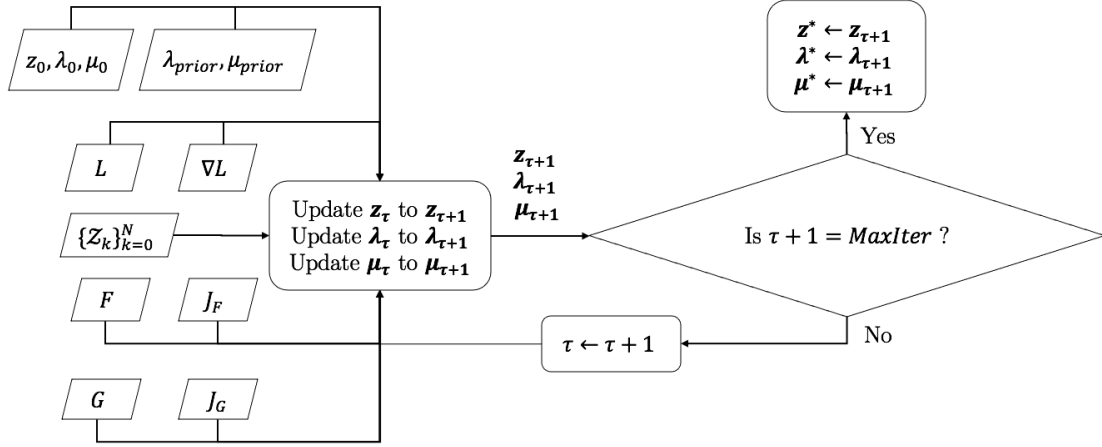


Figure 3.3: Primal-Dual Online Optimization Flowchart

$L_{k_0}(\mathbf{z}_{k_0})$  is the horizon cost function at the  $k_0^{\text{th}}$  instance of the MPC run

$G_{k_0}(\mathbf{z}_{k_0})$  is the set of explicit inequality constraint functions at the  $k_0^{\text{th}}$  instance of the MPC run

$F_{k_0}(\mathbf{z}_{k_0})$  is the set of explicit equality constraint functions at the  $k_0^{\text{th}}$  instance of the MPC run

### Tracking Error

As a measure of the tracking performance of an online optimization algorithm, we define the *tracking-error* [3.23]. This represents the running distance between the *true optimal* solution  $\mathbf{q}_{k_0}^*$  achieved by running the algorithm to convergence in the batch approach and the *sub-optimal* solution  $\hat{\mathbf{q}}_{k_0}$  generated by truncating the number of iterations in the online approach for absolute time-steps  $k_0 = 0, 1, \dots, T - 1$ .

$$\|\hat{\mathbf{q}}_{k_0} - \mathbf{q}_{k_0}^*\|_\eta = \sqrt{\|\hat{\mathbf{x}}_{k_0} - \mathbf{x}_{k_0}^*\|^2 + \eta^{-1} \left( \|\hat{\lambda}_{k_0} - \lambda_{k_0}^*\|^2 + \|\hat{\mu}_{k_0} - \mu_{k_0}^*\|^2 \right)} \quad (3.23)$$

While it is a given that the running solution will be sub-optimal and thus have a non-zero tracking error, the objective is to *bound* this tracking error.

### Tracking Error Bound

We define the *temporal variability* of the optimization algorithm (adapted to the discrete-time case from [27, 28]) as in [3.24]

$$\sigma_\eta := \sup_{\substack{k_{0,1}, k_{0,2} \in [0, 1, \dots, T-1] \\ k_{0,1} \neq k_{0,2}}} \frac{\|\mathbf{q}_{k_{0,2}}^* - \mathbf{q}_{k_{0,1}}^*\|_\eta}{|k_{0,2} - k_{0,1}|} \quad (3.24)$$

This quantity can be intuitively thought of as the maximum speed with which the KKT point  $\mathbf{q}^*$  changes with respect to the norm  $\|\cdot\|_\eta$ .

We also define the necessary upper-bound quantities in [3.25] - [3.27]

$$M_{\lambda\mu} := \sup_{k_0 \in [0, 1, \dots, T-1]} \left\| \begin{bmatrix} \boldsymbol{\lambda}_{k_0}^* - \boldsymbol{\lambda}_{\text{prior}} \\ \boldsymbol{\mu}_{k_0}^* - \boldsymbol{\mu}_{\text{prior}} \end{bmatrix} \right\| \quad (3.25)$$

$$L_{GF}(\delta) := \sup_{k_0 \in [0, 1, \dots, T-1]} \sup_{\boldsymbol{\Delta z}: \|\boldsymbol{\Delta z}\| \leq \delta} \left\| \begin{bmatrix} J_{G_{k_0, \mathbf{z}_{k_0}}}(\mathbf{z}_{k_0}^* + \boldsymbol{\Delta z}) \\ J_{F_{k_0, \mathbf{z}_{k_0}}}(\mathbf{z}_{k_0}^* + \boldsymbol{\Delta z}) \end{bmatrix} \right\| \quad (3.26)$$

$$D(\delta, \eta) := \sqrt{\eta} L_{GF}(\delta) \quad (3.27)$$

and a quantity to measure the upper-bound nonlinearity of the cost and equality constraint functions with respect to  $\mathbf{z}_{k_0}$  in [3.28](#)

$$M_L(\delta) := \sup_{k_0 \in [0, 1, \dots, T-1]} \sup_{\boldsymbol{\Delta z}: \|\boldsymbol{\Delta z}\| \leq \delta} \|D_{zz}^2 L_{k_0}(\mathbf{z}_{k_0}^* + \mathbf{z})\| \quad (3.28)$$

where:

The bilinear map that maps a pair of vectors  $(h_1, h_2)$  to a vector whose  $i^{\text{th}}$  entry is denoted by  $h_2^T \Delta_{zz}^2 f_i(\mathbf{z}_{k_0}) h_1$  is given by:

$$D_{zz}^2 f_{k_0}(\mathbf{z}_{k_0} : (h_1, h_2)) \mapsto (h_2^T \nabla_{zz}^2 f_i(\mathbf{z}_{k_0}) h_1)_i.$$

$$\|D_{zz}^2 f_{k_0}(\mathbf{z}_{k_0})\| := \sup_{h_1, h_2 \neq 0} \frac{\|D_{zz}^2 f_{k_0}(\mathbf{z}_{k_0})(h_1, h_2)\|}{\|h_1\| \|h_2\|} = \sup_{\|h_1\| = \|h_2\| = 1} \|D_{zz}^2 f_{k_0}(\mathbf{z}_{k_0})(h_1, h_2)\|$$

We also define the following quantities:

$$\bar{H}_{\mathcal{L}, k_0}(\boldsymbol{\Delta z}) := \int_0^1 \nabla_{zz}^2 \mathcal{L}_{k_0}(\mathbf{z}_{k_0}^* + \theta \boldsymbol{\Delta z}, \boldsymbol{\lambda}_{k_0}^*, \boldsymbol{\mu}_{k_0}^*) d\theta \quad (3.29)$$

$$\rho^{(P)}(\delta, \alpha, \eta, \epsilon) := \sup_{\substack{k_0 \in [0, 1, \dots, T] \\ \boldsymbol{\Delta z}: \|\boldsymbol{\Delta z}\| \leq \delta}} \left\| (I - \alpha \bar{H}_{\mathcal{L}, k}(\boldsymbol{\Delta z}))^2 \right\| \quad (3.30)$$

$$\rho(\delta, \alpha, \eta, \epsilon) := \left[ \max \left\{ \rho^{(P)}(\delta, \alpha, \eta, \epsilon), (1 - \eta\alpha\epsilon)^2 \right\} + \alpha(1 - \eta\alpha\epsilon) \frac{\sqrt{\eta}\delta M_L(\delta)}{2} \right. \\ \left. \alpha^2 \left( 2 \sup_{\substack{k_0 \in [0, 1, \dots, T] \\ \boldsymbol{\Delta z}: \|\boldsymbol{\Delta z}\| \leq \delta}} \|\eta\epsilon I - \bar{H}_{\mathcal{L}, k}(\boldsymbol{\Delta z})\| D(\delta, \eta) + D^2(\delta, \eta) \right) \right]^{1/2} \quad (3.31)$$

$$\kappa(\delta, \alpha, \eta, \epsilon) := \max \left\{ 1, \frac{1 - \eta\alpha\epsilon}{\rho(\delta, \alpha, \eta, \epsilon)}, \frac{\sqrt{\eta}\alpha L_{GF}(\delta)}{\rho(\delta, \alpha, \eta, \epsilon)} \right\} \quad (3.32)$$

Theorem [3.3.1](#) establishes the tracking error bound of the implemented regularized primal-dual gradient algorithm.

**Theorem 3.3.1** ([\[27\]](#) Theorem 3.2, [\[28\]](#) Theorem 2.1). *Suppose there exists  $\delta > 0$ ,  $\alpha > 0$ ,  $\eta > 0$  and  $\epsilon > 0$  such that:*

$$\sigma_\eta \leq (1 - \rho(\delta, \alpha, \eta, \epsilon))\delta - \kappa(\delta, \alpha, \eta, \epsilon)\sqrt{\eta}\alpha\epsilon M_{\lambda\mu} \quad (3.33)$$

Let the initial point  $\hat{\mathbf{q}}_0 = (\hat{\mathbf{x}}_0, \hat{\boldsymbol{\lambda}}_0, \hat{\boldsymbol{\mu}}_0)$  be sufficiently close to the KKT point  $\mathbf{q}_1^*$  such that:

$$\|\hat{\mathbf{q}}_0 - \mathbf{q}_1^*\|_\eta \leq \delta. \quad (3.34)$$

Then the sequence  $[\hat{\mathbf{q}}_{k_0}]_{k_0=0}^{T-1}$  generated by the regularized primal-dual gradient algorithm [3.21](#) satisfies:

$$\|\hat{\mathbf{q}}_{k_0} - \mathbf{q}_{k_0}^*\|_\eta \leq \frac{\rho(\delta, \alpha, \eta, \epsilon)\sigma_\eta + \kappa(\delta, \alpha, \eta, \epsilon)\sqrt{\eta}\alpha\epsilon M_{\lambda\mu}}{1 - \rho(\delta, \alpha, \eta, \epsilon)} \\ + \rho^{k_0}(\delta, \alpha, \eta, \epsilon) \left( \|\hat{\mathbf{q}}_0 - \mathbf{q}_1^*\|_\eta - \frac{\sigma_\eta + \kappa(\delta, \alpha, \eta, \epsilon)\sqrt{\eta}\alpha\epsilon M_{\lambda\mu}}{1 - \rho(\delta, \alpha, \eta, \epsilon)} \right) \quad (3.35)$$



for all  $k_0 \in [0, 1, \dots, T - 1]$ . Moreover, we have:

$$\lim_{\alpha \rightarrow 0^+} \kappa(\delta, \alpha, \eta, \epsilon) = 1 \text{ and } \kappa(\delta, \alpha, \eta, \epsilon) \leq \sqrt{2} \quad (3.36)$$

Intuitively speaking, we can say that the slower  $\mathbf{q}_{k_0}^*$  moves over the course of the control trajectory, the more likely it is to achieve a smaller tracking error. The proof of Theorem 3.3.1 is based on Lemma 3.3.2, which establishes Q-linear convergence (defined in Appendix C.4) for the iterative algorithm given by 3.21. We refer the reader to [27, 28] for the full proofs.

**Lemma 3.3.2** ([27] Lemma 3.3, [28] Lemma 2.3). *Let  $k_0 \in \{0, 1, \dots, T - 1\}$  be an arbitrary instance in controller execution time. If  $\hat{\mathbf{q}}_{k_0}$  is generated by 3.21, then:*

$$\|\hat{\mathbf{q}}_{k_0} - \mathbf{q}_{k_0}^*\|_\eta \leq \rho(\delta, \alpha, \eta, \epsilon) \|\hat{\mathbf{q}}_{k_0-1} - \mathbf{q}_{k_0-1}^*\|_\eta + \kappa(\delta, \alpha, \eta, \epsilon) \sqrt{\eta} \alpha \epsilon M_{\lambda\mu} \quad (3.37)$$

Under Assumptions 3.2.7, the tracking error bounds in Theorem 3.3.1 still apply to our case, in which the KKT point trajectory is updated at discrete time-steps, the cost function is considered to be non-convex and  $\text{MaxIter} \geq 1$  iterations are allowed for each MPC problem to be solved. Furthermore, in this work we implement the proposed algorithm in the context of real-time MPC for building control (Sec. 4) and the inverted pendulum problem (Appendix A).

### 3.3.2 Warm-Starting

One approach to promoting fast convergence of the MPC problem is to transfer the solution from the problem solved during the preceding time-step to the subsequent one. *Warm-starting* (related to the concepts of *time-distributed optimization* [21] and *real-time iteration* [22]) involves using an existing solution approximation as an initial guess for the subsequent MPC optimization problem. Using this method, it may be possible to track an optimal solution and eventually converge to it, given very few iterations (or only a single one at the limit) of the optimization algorithm. It has been used successfully in the building control context in [9] and is explained further in [44] Sec 2.7 and 8.9.

The most basic implementation of warm-starting involves setting the initial iterates to the solution of the last MPC problem:

$$\hat{\mathbf{q}}_{k_0+1, \tau=0} := [\hat{\mathbf{q}}_{k_0, k=0}^T, \dots, \hat{\mathbf{q}}_{k_0, k=N}^T]^T \quad (3.38)$$

Alternatively, the *shift-initialization* approach involves shifting the solution back by one time-step to account for the advancement in time, and appending zero values to the end. In this work however, we wish to achieve convergence in as few as a single iteration, and so the terminal stage elements of the initial KKT point are set to the penultimate stage solution:

$$\hat{\mathbf{q}}_{k_0+1, \tau=0} := [\hat{\mathbf{q}}_{k_0, k=1}^T, \dots, \hat{\mathbf{q}}_{k_0, k=N}^T, \hat{\mathbf{q}}_{k_0, k=N}^T]^T \quad (3.39)$$

Shift-initialization is applicable to problems in which an equidistant temporal-grid is used for the MPC time-step and for the dynamic state equation discretization, and is particularly advantageous for time-varying systems. For these reasons we employ it in the real-time control scheme proposed in this work.

## 3.4 Sampling Timeline

There are multiple *sampling time intervals* to consider in Algorithm 3 (illustrated in Fig. 3.4):

- The time-intervals at which the MPC must be executed,  $\Delta t$ . This is directly related to the time-intervals at which optimal or sub-optimal control inputs must be applied to the system and the shortest time-scale at which the functions of the problem vary. In real-time MPC, this time-interval is typically shorter than is required for the solution to converge.

- The time-intervals at which the user feedback is received and added to the online subset of the training data of the GP-learned cost functions,  $\Delta t_l^d$ , such that sufficiently up-to-date and accurate approximations are maintained.
- The time-intervals at which the state feedback is received and added to the online subset of the training data of the GP-learned system models,  $\Delta t_x^d$ , such that sufficiently up-to-date and accurate approximations are maintained.

Consider the case when a real-time controller is required to generate optimal control inputs at regular discrete intervals in order to respond to new environmental disturbances and time-varying objectives. In sampling for the user dissatisfaction GPs, it is burdensome for the user of a device to be frequently queried for feedback. In sampling for the device dynamic state GPs, while it may require considerable cost or effort to take a dynamic system offline to test for new training data, training data based on real-time inputs and outputs can be collected during controller operation. In practice then, the MPC sampling time-interval  $\Delta t$  can be relatively short i.e. on the scale of seconds to minutes, whereas the user dissatisfaction cost GP sampling time-interval  $\Delta t_l^d$  can be relatively long i.e. on the scale of hours to days and the device dynamic state GP sampling time-interval  $\Delta t_x^d$  can be equivalent to  $\Delta t$  (but not necessarily, especially if input-output data is required which is not available at this sampling interval).

It may be the case that periodic offline tests can be conducted to generate a completely new training set of  $N_{tr,0}$  samples, such that once the MPC is restarted, the GP-learned functions have been initialized with new training sets. In our application, this is the case when we initially launch the MPC - the GPs are *pre-trained* with  $N_{tr,0}$  data points before the simulation is begun. Note that  $N_{tr,0}$  may vary for different GPs i.e. quadratic or cosine functions require considerably fewer data points to provide a good approximation compared to other function classes.

Given an initial training set, at each subsequent sampling interval  $\geq 1$  additional data points are received from the user or device. These new data points are appended to the online subset of the training set until it has reached the maximum size of  $N_{tr}$  data points. At this point, the oldest data points in the online subset (the offline subset is left intact) are replaced by the newest such that the size of the training set remains constant. With this *sliding-window* sampling technique, the computational burden of inverting the covariance matrix does not become unbounded with an increasing number of training samples, the training data is kept up-to-date with recent data trends and the GP-learned functions are adapted while the controller is running.

Natural cycles in the system data can be considered when designing the number of training samples  $N_{tr}$ . For example, in one application of this work, the dynamics of a number of TCL devices are considered (Chapter 4). The outdoor temperature  $T_o$ , which exhibits an approximate periodic variation over the time-scale of 1 day, has a significant influence on the zone temperature  $T^d$ , which the MPC is attempting to steer towards a given reference  $T_{ref}^d$ . We can therefore set the initial size of the training set for these dynamic state functions  $N_{tr,0}$  to the equivalent of 1 day (288 samples for data points that are available at 5 minute intervals) such that a wide range of the typical daily variations are captured by the offline subset, and the maximum size of the training set  $N_{tr}$  to the equivalent of 1.5 days (432 samples) such that half a day of training data belongs to the online subset and is constantly evolving based on the system trajectory.

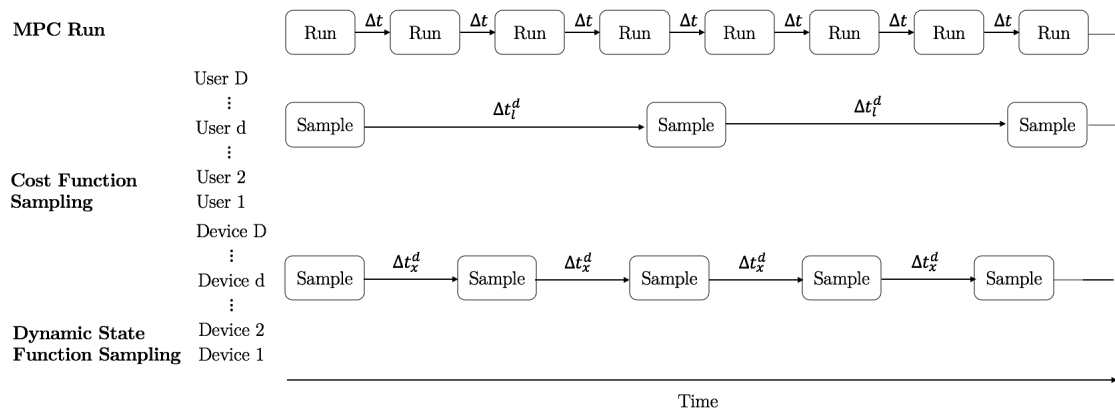


Figure 3.4: Illustration of Asynchronous GP Training Data Collection and MPC Sampling Timelines

## Chapter 4

# Application to Real-Time Optimization of Energy Systems

In this chapter, we implement the proposed MPC scheme to control a network of distributed energy devices in a real-time setting, where the device-specific dynamics and the user-specific cost functions are learned from real-time feedback using Gaussian Processes. All Python code written for the experiments conducted in this chapter can be found at [51].

Consider a *multiuser problem* [12] which, for our purposes, is a *time-varying constrained optimization problem* characterized by:

- a set of *devices* and corresponding *users*,
- feasible sets for the states and inputs of each device,
- a cost function given by the sum of user-specific *dissatisfaction functions* and an aggregate cost term based on a coupling metric,
- a collection of convex inequality constraints coupling the device states and inputs.

In our case, the dynamics of each device and the cost incurred on each user are unknown functions which we need to learn in real-time using GPs. For the purposes of implementing the primal-dual gradient-based optimization method described in Sec. 3.2 - 3.3 we let the primal variables  $\mathbf{z}$  represent the states and inputs of all devices, the dual inequality variables  $\boldsymbol{\lambda}$  correspond to the convex inequality constraints  $G$  and the dual equality variables  $\boldsymbol{\mu}$  correspond to the dynamic state equality constraints  $F$ . The states and inputs of all devices are coupled by the metric  $\mathbf{y}$  which has an associated inequality constraint  $\mathbf{g}^0(\mathbf{y})$  (included in  $G$ ) and known system stage cost  $l^0(\mathbf{y})$ .

### 4.1 System Model

Consider an aggregation of devices in a neighborhood or community which consists of  $D$  *controllable devices* and  $W$  *uncontrollable devices*, similar to the case studies in [13, 14, 29]. In our case the controllable devices consist of  $C$  *thermostatically-controlled load* (TCL) cooling devices (or *HVACs*). The states and inputs of each device  $d$  are constrained by the feasible sets  $\mathcal{X}_k^d$  and  $\mathcal{U}_k^d$  at each time-step  $k$ , respectively. The *total active power output* at the point of interconnection with the rest of the grid at time-step  $k$  is denoted by  $y_k$ . In designing a controller for this system, the dual objectives are:

- a) to track the reference total active power output  $y_{ref,k}$  and
- b) to minimize the user dissatisfaction with each device's operation.

In this section, we will describe the model of the TCL devices employed and the overall network model.

### 4.1.1 Thermostatically-Controlled Load

*Thermostatically-controlled loads* are devices for which the setpoints equate to temperatures (e.g. fridges, freezers, air-conditioners, hot-water tanks, heat pumps, and swimming pool pumps). In this section, we will describe their associated states, control inputs, disturbances and parameters for the purposes of implementation in a building network.

#### States

The states of TCL  $d$  at each time-step  $k$  consist of the zone temperature,  $T_k^d$ .

$$\mathbf{x}_k^d := [T_k^d [^\circ C]] \forall k = 0, 1, \dots, N \quad (4.1a)$$

$$\mathcal{X}_k^d := [\underline{T}_k^d, \overline{T}_k^d] \forall k = 0, 1, \dots, N \quad (4.1b)$$

$$\mathcal{X}_f^d := [\underline{T}_N^d, \overline{T}_N^d] \quad (4.1c)$$

where:

$\overline{T}_k^d := \min \left( \max (X^d[T_k^d]), \overline{T}_k^d \right)$  is the time-varying maximum allowed zone temperature

$\underline{T}_k^d := \max \left( \min (X^d[T_k^d]), \underline{T}_k^d \right)$  is the time-varying minimum allowed zone temperature

Note that we constrain the zone temperature to be the intersection of the nominal feasible sets (given by the upper and lower bounds,  $\overline{T}_k^d$  and  $\underline{T}_k^d$ ) and the zone temperature in the available training data  $X^d[T_k^d]$  in order to steer the system trajectory to regions which are explored (at least for some dimensions) by the GPs.

#### Control Inputs

The control inputs of the TCL at each time-step consist of the total cooling power produced,  $P_k^d$ . Note that this will always be nonpositive, as the TCL consumes power to cool its zone.

$$\mathbf{u}_k^d := [P_k^d [\text{kW}]] \forall k = 0, 1, \dots, N - 1 \quad (4.2a)$$

$$\mathcal{U}_k^d := [-\overline{P}_k^d, -\underline{P}_k^d] \forall k = 0, 1, \dots, N - 1 \quad (4.2b)$$

where:

$\overline{P}_k^d := \min \left( \max (X^d[P_k^d]), \overline{P}_k^d \right)$  is the time-varying maximum allowed power consumed

$\underline{P}_k^d := \max \left( \min (X^d[P_k^d]), \underline{P}_k^d \right)$  is the time-varying minimum allowed power consumed

Similar to the zone temperature, we constrain the cooling power to be in the intersection of the nominal feasible sets (given by the upper and lower bounds,  $-\underline{P}_k^d$  and  $-\overline{P}_k^d$ ) and the cooling power in the available training data  $X^d[P_k^d]$ .

The closed-form expressions for this quantity are given by [4.3](#)

$$P_{fan,k}^d = 0.0076 (\dot{m}_k^{total})^3 + 4.8865, \quad \dot{m}_k^{total} = \sum_{d=1}^D \dot{m}_k^d \quad (4.3a)$$

$$P_{ch,k}^d = \begin{cases} \dot{m}_k^{total} (T_{o,k}^d - T_{da,k}^d) & \text{if } T_{o,k} > T_{da,k}^d \\ 0 & \text{otherwise} \end{cases} \quad (4.3b)$$

$$P_k^d = -(P_{fan,k}^d + P_{ch,k}^d) \quad (4.3c)$$

where:

$P_{fan}^d$  is the power consumed by the fan

$P_{ch}^d$  is the power consumed by the chiller

$\dot{m}_a^d$  is mass flow air

$\dot{m}_a^{total}$  is total mass flow air over all TCLs

$T_{da}^d$  is the discharge air temperature

$T_o$  is the outdoor air temperature

$Q_s^d$  is the solar irradiance

$Q_i^d$  is the internal irradiance

### Disturbances

The disturbances of the system consist of the temperature outside of the building at each time-step,  $T_{o,k}$ . Note that this disturbance is common to all TCL devices in the network.

$$\mathbf{w}_k^d := [T_{o,k} [^\circ C]] \forall k = 0, 1, \dots, N - 1 \quad (4.4)$$

### Dynamic State Equation

The discrete dynamic state equation of the TCL is assumed to be unknown. Instead of considering available analytical models, we will learn the relationship between the current zone temperature, cooling power input, outdoor temperature and the zone temperature in the subsequent time-step using GPs trained with real data [52]. This approach can be advantageous because both the model and the parameterization used in analytical models may be system- or time-dependent. By relying on a model learned online, we can learn to control the system in question in real-time, without the error introduced by the synthetic model. The prior state in this case is set to zero, and we use the GP to learn the entire next state, see [4.5].

$$T_{k+1}^d := \underbrace{\bar{g}(T_k^d, P_k^d, T_{o,k})}_{\text{GP posterior mean of next state}} \quad (4.5)$$

The *true* system must also be modeled, such that once each MPC run terminates, the resulting control input can be applied to the real system and the true next state obtained. In our case, we do not have access to the real building system, and thus we model the true system with a GP which uses several days of training data and has a prior mean equal to its reference temperature (which we choose to approximately equal the mean of the training data). The result is a GP-modeled true system which is computationally burdensome but precise enough for our purposes. This provides a safe system output in the case that the GP inputs are in an unexplored region.

### Device Stage Cost Function

The time-varying stage cost function of the TCL user is also unknown. We synthesize noisy samples from a given function and approximate this function with a GP. We take the stage cost prior to be 0 (i.e. we have no full knowledge of particular terms in the function) and thus learn the entire relationship with the GP, see [4.6](#).

$$l^d(T_k^d, P_k^d, T_o) := \underbrace{\bar{j}(T_k^d, P_k^d, T_{o,k})}_{\text{GP posterior mean of stage cost}} \quad (4.6)$$

The true stage cost function is based on the squared difference between the temperature and the temperature reference at each time step  $k$ , see [4.7](#).

$$l^{\text{true},d}(T_k^d, P_k^d, T_o) = \underbrace{(T_k^d - T_{k,\text{ref}}^d)^2}_{\text{true stage cost}} \quad (4.7)$$

### Parameters

The parameters provided to the TCL consist of the temperature reference  $T_{ref,k}^d$ , the nominal maximum zone temperature  $\bar{T}_k^d$ , the nominal minimum zone temperature  $\underline{T}_k^d$ , the nominal maximum cooling power used  $\bar{P}_k^d$  and the nominal minimum cooling power used  $\underline{P}_k^d$  at time-step  $k$ , see [4.8](#).

$$\mathbf{p}_k := \begin{bmatrix} T_{ref,k}^d \\ \bar{T}_k^d \\ \underline{T}_k^d \\ \bar{P}_k^d \\ \underline{P}_k^d \end{bmatrix} \quad (4.8)$$

#### 4.1.2 Network

##### States

The network states include the  $n_x^d$  measurable states  $\mathbf{x}_k^d$  of each device  $d$  at each time-step  $k$  e.g. state-of-charge (SOC) in the case of an energy storage system (ESS) (not implemented in this work) or zone temperature in the case of a HVAC for all devices and time-steps. Each device's state vector has an associated feasible set  $\mathcal{X}_k^d$ . The states and corresponding feasible sets of all devices [\(4.9a\)](#) are combined into aggregate state vectors for each time-step  $k$  [\(4.9b\)](#). The states and corresponding feasible sets of all time-steps are combined into a single aggregate state vector and feasible set [\(4.9c\)](#), generating a single vector of states which can be used in an optimization algorithm.

$$\mathbf{x}_k^d \in \mathcal{X}_k^d \subseteq \mathbb{R}^{n_x^d} \quad (4.9a)$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^1 \\ \vdots \\ \mathbf{x}_k^D \end{bmatrix} \in \mathcal{X}_k := \mathcal{X}_k^1 \times \cdots \times \mathcal{X}_k^D \subseteq \mathbb{R}^{\sum_{d=1}^D n_x^d} \quad (4.9b)$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \in \mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_N \subseteq \mathbb{R}^{N \sum_{d=1}^D n_x^d} \quad (4.9c)$$

### Control Inputs

Control inputs include the *active power output setpoints* dispatched to each device  $d$  at each time-step  $k$  (4.10a). These control input vectors and their associated feasible sets are aggregated into a control input vector and feasible set for each time-step (4.10b), which are in turn aggregated into a single vector and feasible set over all time-steps (4.10c), which can be used in an optimization algorithm.

$$\mathbf{u}_k^d \in \mathcal{U}_k^d \subseteq \mathbb{R}^{n_u^d} \quad (4.10a)$$

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_k^1 \\ \mathbf{u}_k^2 \\ \vdots \\ \mathbf{u}_k^D \end{bmatrix} \in \mathcal{U}_k := \mathcal{U}_k^1 \times \mathcal{U}_k^2 \times \cdots \times \mathcal{U}_k^D \subseteq \mathbb{R}^{\sum_{d=1}^D n_u^d} \quad (4.10b)$$

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix} \in \mathcal{U} := \mathcal{U}_0 \times \mathcal{U}_1 \times \cdots \times \mathcal{U}_{N-1} \subseteq \mathbb{R}^{N \sum_{d=1}^D n_u^d} \quad (4.10c)$$

If commands are dispatched to device  $d$  at a slower rate of every  $i + j$  time-steps for  $j > 0$ , the most recently issued command  $\mathbf{u}_k^d$  is held constant until the next command  $\mathbf{u}_{i+j}^d$  is received. In this case  $\mathcal{U}_k^d, k = i + 1, \dots, j - 1$  would be a singleton set while the command is held constant.

### Disturbances

The disturbances of the system consist of the power outputs for each of the uncontrolled loads  $d$  in the building network at each time-step  $k$ ,  $w_k^d$ . These disturbances are aggregated into a vector corresponding to each time-step (4.11) and subsequently into a single vector of disturbances (4.12) which can be passed as a parameter to an optimization algorithm.

$$\mathbf{w}_k = \begin{bmatrix} w_k^1 \\ w_k^2 \\ \vdots \\ w_k^W \end{bmatrix} \quad (4.11)$$

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_{N-1} \end{bmatrix} \subseteq \mathbb{R}^{N \sum_{d=1}^W n_w^d} \quad (4.12)$$

### Outputs

The controlled setpoints  $\mathbf{u}_k$  and the uncontrolled power outputs  $\mathbf{w}_k$  are mapped to the measured *net real electrical power* produced by the network at time-step  $k$ ,  $y_k \in \mathbb{R}$ , by the function  $y_k : \mathbb{R}^D \times \mathbb{R}^W \mapsto \mathbb{R}$  (4.13) which takes the controlled setpoints as an input and the uncontrolled power outputs as a parameter.

$$y_k(\mathbf{u}_k; \mathbf{w}_k) := \mathbf{1}^{1 \times D} \mathbf{u}_k + \mathbf{1}^{1 \times W} \mathbf{w}_k \quad (4.13)$$

The coupling metric (4.13) is representative of the case in which we assume all devices are connected to a common electrical node e.g. a building or collection of buildings with a common distribution feeder.



### GP-Modeled Functions

There are a number of unknown functions in our system which must be learned concurrently with the execution of the controller using GPs:

- The (possibly nonlinear, non-convex) user dissatisfaction stage cost function of each device  $d$ ,  $l^d$  may be unknown, outdated or deviate greatly from approximated synthetic models. We will model these functions with GPs and use their posterior means  $\bar{l}^d$  and closed-form expressions for the derivatives of the posterior means  $\Delta\bar{l}^d$  given feedback collected from the device users *in lieu* of closed-form expressions.
- The (possibly nonlinear, non-convex) dynamics of each state  $i$  of each device  $d$ ,  $f_i^d$  may be unknown. We will model these functions with GPs and use their posterior means  $\bar{f}_i^d$  and closed-form expressions for the derivatives of the posterior means  $\Delta\bar{f}_i^d$  given feedback collected from the true device system, *in lieu* of closed-form expressions.

Additionally, it may not be possible to collect reliable power output measurements from the uncontrollable loads  $\mathbf{w}_k$  and it may therefore be impossible to directly calculate the total active power  $y_k$  or to compute its gradient at each time-step  $k$ . In the case that a single iteration is allowed for the algorithm to converge at each time-step, we could use the measurement of the active power at the point of interconnection  $\hat{y}_k$  in lieu of the deterministic expression given by [4.13](#) and employ the measurements to approximate the gradient by the finite-difference method [2.25](#).

### Network Performance Stage Cost Function

The *objective* of this case-study is to formulate a demand-side management problem (similar to [13](#), [14](#)) which allows real-time scheduling of end-user devices by minimizing a cost function that considers both user satisfaction and network performance over a horizon of time-steps projected into the future.

The MPC cost function accordingly includes terms represented by unknown *dissatisfaction functions*  $l^d(\mathbf{z}_k^d)$  corresponding to the user(s) of each device  $d$ . For example, in the case of TCLs the discomfort function may model the discomfort of the user for deviations from a preferred temperature setpoint, or in the case of ESSs the discomfort function may model the dissatisfaction of the user for deviations from a preferred charging profile. In this work, these discomfort functions are unknown *a priori* and are subsequently learned from user feedback using GPs.

The cost function also includes a known time-varying *performance* or *engineering cost* which couples the states and setpoints of all devices,  $l_k^0(y_k)$ . In our case this is based on the squared difference between the net power output of the system and a preferred power output profile, see [4.14](#).

$$l_k^0(y_k) := \frac{\beta}{2} \|y_k - y_{ref,k}\|_2^2 \quad (4.14)$$

The overall network model is illustrated in Fig. [4.1](#). In theory, the framework presented in this chapter could include a number of ESSs, for example electric vehicle (EV) batteries connected to the building power network. While multiple models and datasets of batteries were tested and analyzed, a source of high-granularity data suitable for GP-modeling could not be obtained. The EV batteries are included in the illustration to demonstrate how a network of power-consuming (TCLs) and sometimes power-producing (EV batteries) devices could be interconnected in a control system which tracks a net power output.

## 4.2 MPC Optimal Control Problem

Model Predictive Control is a well-established method for real-time control of a network of distributed energy systems, as shown in [9](#). In order to control the states of each device in the

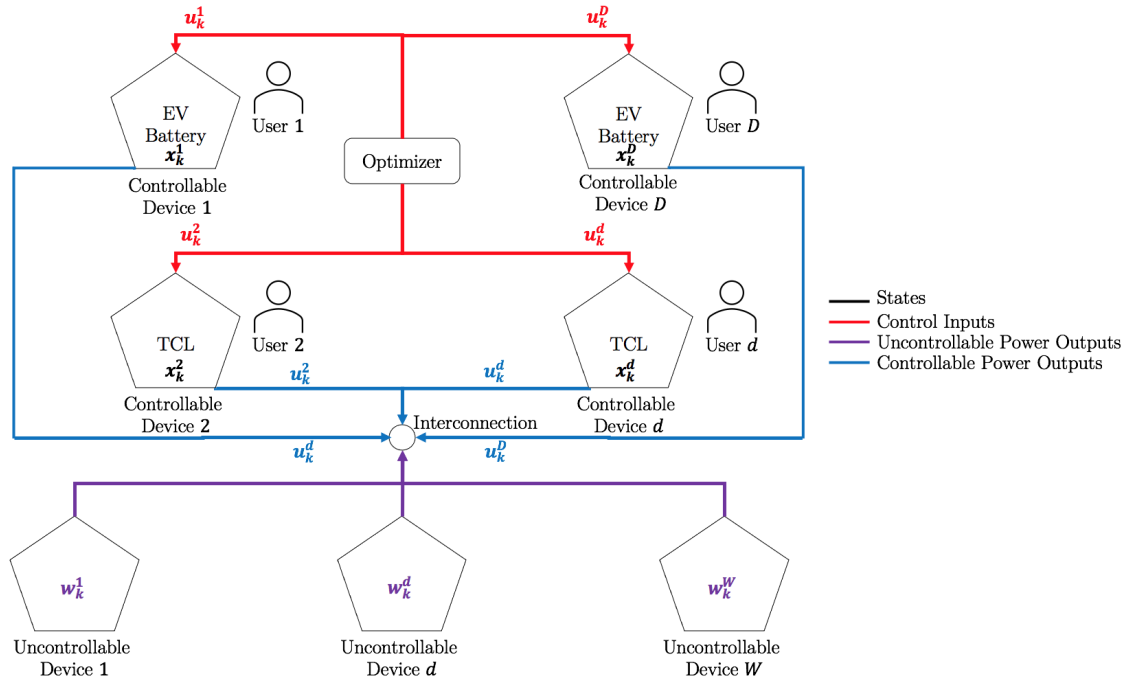


Figure 4.1: A Network of  $D$  Controllable Devices and  $W$  Uncontrollable Devices, where  $\mathbf{u}_k^d$  represents a control input signal distributed from the centralized operator to device  $d$  at time-step  $k$  and subsequently output in the form of power to a common interconnection,  $\mathbf{x}_k^d$  represents the state vector of device  $d$  at time-step  $k$  and  $\mathbf{w}_k^d$  represents the power output from uncontrollable device  $d$  at time-step  $k$ .

network given constraints coupling the states and inputs of multiple devices, a cost coupling the states and inputs of multiple device, feasible sets for the states and control inputs of each device and user dissatisfaction costs for each device we can formulate a multiuser MPC optimization control problem.

As described in Sec. 4.1.2, we have a networked system consisting of a finite set of  $D$  controllable devices with corresponding users and  $W$  uncontrollable devices. The states  $\mathbf{x}_k^d$  and control inputs  $\mathbf{u}_k^d$  of each controllable device  $d$  at each time-step  $k$  are constrained by the time-varying feasible sets  $\mathcal{X}_k^d$  and  $\mathcal{U}_k^d$ , respectively. The horizon cost function  $L(\{\mathbf{z}_k\}_{k=0}^N)$  captures both the separable *device cost functions*  $l^d(\mathbf{z}_k^d)$  for the controllable devices that depend only on that device's decision variables, and a *system cost function*  $l^0(y_k)$  which couples the cost of the individual controllable devices' decision variables. We combine the state feasible sets  $\mathcal{X}_k^d$  and the control input feasible sets  $\mathcal{U}_k^d$  into the time-varying decision variable feasible set  $\mathcal{Z}_k^d$ . For computational purposes, we reformulate the dynamic state equations as equality constraints  $F_{k,i}^d(\mathbf{z}_k^d) = x_{k+1,i}^d - \bar{f}_i^d(\mathbf{z}_k^d) = 0$ . The multiuser discrete-time MPC with GPs can thus be formulated as in 4.15.

$$\min_{\{\mathbf{z}_k\}_{k=0}^N} L(\{\mathbf{z}_k\}_{k=0}^N) := \sum_{k=0}^N \left[ \sum_{d=1}^D \underbrace{\bar{l}^d(\mathbf{z}_k)}_{\text{unknown stage user cost}} \right] + \underbrace{l_k^0(y_k)}_{\text{known stage system cost}} \quad (4.15a)$$

$$\text{s.t. } \mathbf{x}_0^d = \mathbf{x}^d(t) \text{ for } d = 1, \dots, D \quad (4.15b)$$

$$F_{k,i}^d(\mathbf{z}_k^d) = 0 \text{ for } d = 1, \dots, D; k = 0, \dots, N-1; i = 1, \dots, n_x^d \quad (4.15c)$$

$$\mathbf{z}_k^d \in \mathcal{Z}_k^d \text{ for } d = 1, \dots, D; k = 0, \dots, N-1 \quad (4.15d)$$

$$\mathbf{x}_N^d \in \mathcal{X}_f^d \text{ for } d = 1, \dots, D \quad (4.15e)$$

$$g_{k,j}^d(\mathbf{x}_k^d) \leq 0 \text{ for } d = 1, \dots, D; k = 0, \dots, N; j = 1, \dots, n_g^d \quad (4.15f)$$

$$g_{k,j}^0(y_k) \leq 0 \text{ for } k = 0, \dots, N; j = 1, \dots, n_g^0 \quad (4.15g)$$

where:

$n_x^d$  is the number of state variables associated with device  $d$

$n_u^d$  is the number of control input variables associated with device  $d$

$n_w^d$  is the number of disturbance variables associated with device  $d$

$n^d = n_x^d + n_u^d + n_w^d$  is the number of decision variables associated with device  $d$

$n_x := \sum_{d=1}^D n_x^d$  is the number of state variables associated with all devices

$n_u := \sum_{d=1}^D n_u^d$  is the number of control input variables associated with all devices

$n_w := \sum_{d=1}^D n_w^d$  is the number of disturbance variables associated with all devices

$n := n_x + n_u$  is the number of decision variables for all devices at each time-step

$n_g^d$  is the number of inequality constraints associated with device  $d$  at each time-step

$n_g^0$  is the number of coupling inequality constraints associated with the network at each time-step

$l_k^0(y_k) : \mathbb{R} \mapsto \mathbb{R} := \frac{\beta}{2} \|y_k - y_{ref,k}\|_2^2$  is a time-varying smooth and convex function specifying costs associated with the system (net power) output  $y_k$  at each time-step  $k$

$y_{ref,k} \in \mathbb{R}$  is a time-varying reference signal for the active power at the point of interconnection

$\bar{l}^d(\mathbf{z}_k^d) : \mathbb{R}^{n^d} \mapsto \mathbb{R}$  is the posterior mean of an unknown, GP-approximated function specifying costs incurred on user  $d$  at each time-step  $k$

$\mathbf{x}^d(t)$  is the true initial state measured from the true device  $d$

$\mathbf{z}_k^d \in \mathbb{R}^{n^d} := [\mathbf{x}_k^{dT} \ \mathbf{u}_k^{dT}]^T$  is the vector of device  $d$  states and inputs. It is the  $d^{\text{th}}$  sub-vector of  $\mathbf{z}_k$ .

$F_{k,i}^d(\mathbf{z}_k^d) : \mathbb{R}^{n^d} \mapsto \mathbb{R} := x_{k+1,i}^d - \bar{f}_i^d(\mathbf{z}_k^d) = 0$  is a function imposing the dynamic state equality constraint on the decision variables of state  $i$  device  $d$  at time-step  $k$

$g_{k,j}^d(\mathbf{x}_k^d) : \mathbb{R}^{n_x^d} \mapsto \mathbb{R}$  is a function imposing time-varying constraints on device  $d$  at time-step  $k$  where we assume that  $g_{k,j}^d(y_k)$  is nonlinear and convex for  $j = 1, \dots, n_{g,n}^d$ , whereas  $g_{k,j}^d(y_k)$  is linear or affine for  $j = n_{g,n}^d + 1, \dots, n_g^d$

$g_{k,j}^0(y_k) : \mathbb{R} \mapsto \mathbb{R}$  is a function imposing time-varying constraints on the system output  $y_k$  at time-step  $k$  where we assume that  $g_{k,j}^0(y_k)$  is nonlinear and convex for  $j = 1, \dots, n_{g,n}^0$ , whereas  $g_{k,j}^0(y_k)$  is linear or affine for  $j = n_{g,n}^0 + 1, \dots, n_g^0$

$\mathcal{X}_k^d \subseteq \mathbb{R}^{n_x^d}$  is the feasible set for the states of device  $d$  at time-step  $k$

$\mathcal{X}_f^d \subseteq \mathbb{R}^{n_x^d}$  is the terminal feasible set for the states of device  $d$

$\mathcal{X}_k := \mathcal{X}_k^1 \times \dots \times \mathcal{X}_k^D$  is the feasible set for the states of all devices at time-step  $k$

$\mathcal{X}_f := \mathcal{X}_f^1 \times \dots \times \mathcal{X}_f^D$  is the terminal feasible set for the states of all devices

$\mathcal{U}_k^d \subseteq \mathbb{R}^{n_u^d}$  is the feasible set for the control inputs of device  $d$  at time-step  $k$

$\mathcal{U}_k := \mathcal{U}_k^1 \times \dots \times \mathcal{U}_k^D$  is the feasible set for the control inputs of all devices at time-step  $k$

$\mathcal{Z}_k := \mathcal{X}_k \times \mathcal{U}_k$  is the feasible set for all decision variables at time-step  $k$

$\mathcal{Z} := \mathcal{Z}_0 \times \dots \times \mathcal{Z}_N$  is the feasible set for all decision variables and all time-steps

For the purposes of implementation with a gradient-based algorithm, we aggregate the variables and constraints over all devices and time-steps into single sets:

$\mathbf{z}_k \in \mathbb{R}^n := [\mathbf{z}_k^1 T \dots \mathbf{z}_k^D T]^T$  is the vector of optimization variables at time-step  $k$

$\mathbf{z} \in \mathbb{R}^{Nn} := [\mathbf{z}_1 T \dots \mathbf{z}_N T]^T$  is the vector of optimization variables over all time-steps

$F_k(\mathbf{z}_k) := [F_{k,1}^1(\mathbf{z}_k^1), \dots, F_{k,n_x^d}^1(\mathbf{z}_k^1), \dots, F_{k,1}^D(\mathbf{z}_k^D), \dots, F_{k,n_x^D}^D(\mathbf{z}_k^D)]^T$  is the set of dynamic state equality constraints at time-step  $k$

$F(\mathbf{z}) := [F_0(\mathbf{z}_0)^T, \dots, F_{N-1}(\mathbf{z}_{N-1})^T]^T$  is the set of dynamic state equality constraints over all time-steps

$\boldsymbol{\mu} \in \mathbb{R}^{Nn_x}$  is the vector of equality dual variables associated with all constraints/rows in  $F$

$G(\mathbf{z}) := [g_0^0(\mathbf{y}_0)^T, \dots, g_N^0(\mathbf{y}_N)^T, g_0^1(\mathbf{x}_0^1)^T, \dots, g_N^1(\mathbf{x}_N^1)^T, \dots, g_0^D(\mathbf{x}_0^D)^T, \dots, g_N^D(\mathbf{x}_N^D)^T]^T$  is the set of explicit output and state constraints over all time-steps

$\boldsymbol{\lambda} \in \mathbb{R}^{N(n_g^0 + n_g)}$  is the vector of inequality dual variables associated with all constraints/rows in  $G$

$L(\mathbf{z}) := \sum_{k=1}^N \sum_{d=1}^D \bar{l}^d(\mathbf{z}_k^d) + \bar{l}^0(y_k)$  is the total horizon cost function

Of particular importance above, are the *optimization variables*,  $\mathbf{z}$ , the *inequality constraints*,  $G(\mathbf{z})$ , the *equality constraints*,  $F(\mathbf{z})$ , the *feasible set*,  $\mathcal{Z}$  of the problem and the *cost function*  $L(\mathbf{z})$ .

The *optimal solution*  $\mathbf{z}^*$  is then the trajectory which minimizes the cost function  $L(\mathbf{z})$  of the system over the next  $N$  time-steps by varying the optimization variables  $\mathbf{z}$  within the implicit and explicit constraints given by  $\mathcal{Z}$ ,  $F(\mathbf{z})$  and  $G(\mathbf{z})$ .

### 4.3 Results of GP Predictions

In this section, we provide relevant parameterization, results and plots of the GP training data, the GP predictions (the posterior mean of the GP at the test inputs), the GP prediction error ( $\pm 1 - 3$  times the standard deviation of the prediction) and scores of the unknown dynamic state and stage cost functions which are modeled by GPs.

### 4.3.1 TCL State Variation

In this section, we use the training data from a single TCL device to test the GP model. We omit the superscripts  $d$  and  $i$  for brevity, as we are considering only a single device and a single state. In order to evaluate the GP score, true values of the outputs are required. We therefore take a window of available data and alternate a given number of training points with a given number of test points, at the fixed ratio  $N_{tr} : N_*$ . There are three input variables to consider (the zone temperature  $T_k$ , the cooling power  $P_k$  and the outdoor temperature  $T_{o,k}$ ) and a single output variable (the subsequent zone temperature  $T_{k+1}$ ). Given that this is too many variables to represent on a two- or three-dimensional plot, we plot the state variation against time to visualize the relationship.

Table 4.1 gives the optimized hyperparameters used (see Sec. 2.5) and the scores calculated (defined in 2.24) and Fig. 4.2 shows the score of the GP model for increasing numbers of training values. Note how the score saturates to a near maximum value of 0.9999 with only 95 training samples. Thereafter, the score varies between a maximum value of 1 and 0.9999 due to numerical rounding errors.

$N_{tr}$	$N_*$	$\mathbf{l}_g$	$\sigma_g^2$	$\sigma_{n,g}^2$	Score
47					0.9995
95					0.9999
143	48	[1.962, 7.729, 7.001]	15.631	$3.8351 * 10^{-4}$	1.0
191					1.0
239					0.9999
287					1.0

Table 4.1: State GP Predictions for  $T_{k+1} = g(T_k, P_k, T_{o,k})$ , where predictions are made at intermittent points extracted in the training data region.

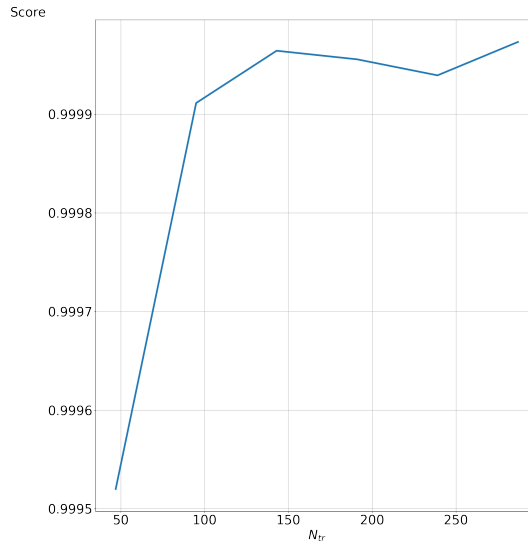


Figure 4.2: State GP Prediction Score vs. No. Training Samples for  $T_{k+1} = g(T_k, P_k, T_{o,k})$ , where predictions are made at intermittent points extracted in the training data region.

Fig. 4.3 compares the training data taken at regular intervals from the real dataset and the GP predictions at the test data points taken at alternating intervals from the real dataset. We see that for any of the training sets tested, the GP can capture all significant variations seen

in the training data, given that it is tested at data points that are close in Euclidean space to the training data.

In contrast, Table 4.2, Fig. 4.5 and Fig. 4.4 give the GP parameterization, scores and predictions, where the training data is taken from a continuous subset of the real dataset and the GP predictions are made over a single hour (12 samples for a sampling time of 5 minutes) of test data points taken from a continuous subset of the real dataset subsequent to the training data. In this case, where the GP is trained and tested over different input regions, we see that the results for 47 – 95 training samples are not sufficient to model the true function. By 143 samples, there is sufficient variation along the range of the training data such that the true function is at least captured by the predicted standard deviation, and by 287 samples major variations in the function are captured quite well by the posterior mean of the prediction. Note how, in the scores given in Table 4.2 and Fig. 4.4, the score does not increase monotonically with the training data. This is because we are training over different input regions for each case, and testing over different regions too. While we cannot fairly compare the different predictions in this case, we can observe how well the GP can predict over a future time horizon (highly relevant considering that it is to be integrated into a MPC receding horizon scheme) given a sliding-window of training data collected and stored online.

$N_{tr}$	$N_*$	$\mathbf{l}_g$	$\sigma_g^2$	$\sigma_{n,g}^2$	Score
47					0.9895
95					0.2279
143	48	[1.962, 7.729, 7.001]	15.631	$3.8351 * 10^{-4}$	0.9997
191					0.9991
239					0.9999
287					0.927

Table 4.2: State GP Predictions for  $T_{k+1} = g(T_k, P_k, T_{o,k})$ , where predictions are made for the hour following the training data.

### 4.3.2 TCL Stage Cost

The stage cost of the TCL is synthesized as the squared difference between the current zone temperature and a preferred zone temperature. As we use synthesized data, we can test the GP *independently* for each input feature, i.e. generate a training set where only one variable is nonzero, and then test with data points where only the same variable is nonzero. For this particular case, it is not necessary, as the true stage cost is only a function of one variable anyway, but it proves a useful testing approach in the case of multivariate functions e.g. in the case of the inverted pendulum in Appendix A.

Table 4.3 gives the manually-set hyperparameters and the scores for different numbers of training samples. In the case of a quadratic function, it is trivial to set the hyperparameters by inspection - the length scale corresponding to a variable upon which the quadratic function is dependent is the domain of that variable and the length scale of any other variable is set to an arbitrarily small value, i.e. we are implementing Automatic Relevance Determination by giving irrelevant dimensions small length scales, such that the covariance associated with these dimensions is negligible. Fig. 4.6 shows the score of the stage cost GP model and Fig. 4.7 shows the training data and the resulting GP predictions of the stage cost function for different numbers of training samples. As we can see, a near perfect quadratic relationship is modeled with only 5 training samples. This exemplifies how optimized hyperparameters result in a GP that becomes incrementally more accurate with an increasing number of training points. The alternative scenario, which can occur for a poorly parameterized covariance kernel, is that the GP attempts to *overfit* the training data given more samples, resulting in higher frequency variation in the prediction than is necessary to model the true function.

$N_{tr}$	$N_*$	$T_{k,ref}$	$\mathbf{l}_j$	$\sigma_j^2$	$\sigma_{n,j}^2$	Score
5						1.0
10	50	24	[8.0, $10^{-6}$ , $10^{-6}$ ]	50.0	$10^{-6}$	1.0
25						1.0
50						1.0

Table 4.3: Stage Cost GP Predictions for  $l^{TCL} = j(T_k, P_k, T_{o,k})$ .

## 4.4 Results of MPC Simulations

A network of 5 TCL devices are simulated over 288 time-steps of 5 minute duration (equivalent to a single day). Refer to Sec. 4.2 for a description of the variables plotted. The TCL, GP and Network parameters are given in Tables 4.4, 4.5 and 4.6, respectively. Note that by formulating the upper and lower bounds of the optimization variables as in Table 4.4, we manually set bounds which are displaced by the upper and lower bounds of the corresponding dimension in the training data if necessary (see the last paragraph of Sec. 2.14 for the reasoning behind this).

$\underline{T}_k^d$	$\bar{T}_k^d$	$\bar{P}_k^d$	$T_{ref,k}^d$
$\max \left( \begin{bmatrix} 22 \\ 21.5 \\ 22.5 \\ 21 \\ 23 \end{bmatrix}, \min(X^d[T^d]) \right)$	$\min \left( \begin{bmatrix} 28 \\ 28.5 \\ 28 \\ 28.5 \\ 27.5 \end{bmatrix}, \max(X^d[T^d]) \right)$	$\max(48, \min(X^d[P^d]))$	$\begin{bmatrix} 24 \\ 23.5 \\ 24.5 \\ 23.5 \\ 24.5 \end{bmatrix}$

Table 4.4: TCL Parameters

Dynamic State/Stage Cost Function	$N_{tr,0}$	$N_{tr}$	$\Delta t_x^d / \Delta t_l^d$ [s]
Dynamic State	287 (1 day)	431 (1.5 days)	60 (5 minutes)
Stage Cost	25	50	3600 (1 hour)

Table 4.5: GP Parameters

Fig. 4.8 shows the convergence of the iterates for the first simulation time-step  $k_0 = 0$ , where Fig. 4.8a shows the trajectory of each primal variable (top plot), each dual variable (middle plot) and the horizon cost function (bottom plot) and Fig. 4.8b shows the L2-norm change in the primal (top plot) and dual (bottom plot) variables, the values of which determine when the gradient algorithm stops converging (more specifically, when the value of the L2-norm of the KKT point, or the combined primal and dual variables, falls below a given threshold). Although the maximum allowed number of iterations in this case is set to 2000, we see that there is negligible variation in the iterates within 80 iterations. This is in part due to the fact that the bounds on the states and inputs are expressed by convex and compact feasible sets as opposed to explicit inequality constraints, and so they have no associated inequality dual variables  $\lambda$  to converge. The fast convergence is also promoted by the constant temperature reference i.e. tracking of a constant reference (except when the power reference changes) results in states that don't need to vary drastically from one time-step to the next. The primal variables  $\mathbf{z}$ , and thus the cost function  $L(\mathbf{z})$  which depends on them, converge directly to their local optima. While there is some overshoot in the dual equality variables  $\mu$ , they too quickly settle to their local optima.

Fig. 4.9 shows the trajectory of the 'true' states  $\mathbf{x}_0$  (recall that we use a highly-trained GP with a prior mean set to the reference temperature as the true system), optimized control inputs  $\mathbf{u}_0$ , horizon cost  $L(\mathbf{z})$  and equality constraint violation  $F(\mathbf{z})$  over a simulation period of 288

$D$	$W$	$\beta$	$y_{ref,k}$
5	0	0.02	Alternating between $-32$ and $-24$ every 72 time-steps

Table 4.6: Network Parameters

Batch/Real-Time	$\Delta t$	$\nu$	$T$	$\alpha$	$\eta$	$\epsilon$	MaxIter	$N$
Batch							2000	6
Real-Time							100	3
Real-Time							100	6
Real-Time							100	12
Real-Time	300 (5 minutes)	$10^{-4}$	288	0.075	1	1	1	6
Real-Time							5	6
Real-Time							10	6
Real-Time							25	6
Real-Time							50	6

Table 4.7: MPC Simulation Parameters

5-minute time-steps (see Sec. 4.2 for a full description of the variables). Note from Table 4.6 that the reference value for the net power output  $y_{ref,k}$  changes every 72 time-steps, and we can see this variation in the horizon cost and in the optimized control inputs. It is evident that one TCL device in particular, that associated with  $\mathbf{x}_{0,3}$ ,  $\mathbf{u}_{0,3}$  and  $F_3$ , is perhaps not modeled as well by its corresponding GP as the other devices. This may be because the trajectory of the simulations is diverging from its explored region of input data and that it requires additional exploratory pre-training before being integrated into the control scheme, or more granular online training. Other features to note are the occasional oscillations in the state trajectories. This is likely because we are using a highly-training GP as the ‘true’ system in our simulations, and so sometimes a cyclical variation in state can occur when an optimal input generated from the MPC using the ‘weakly-trained’ GPs is passed to an equivalent ‘highly-trained’ GP to fetch the true next state, which is subsequently fed back to the ‘weakly-trained’ GPs in the next MPC run.

Fig. 4.10 shows the simulation trajectories for a fixed maximum number of iterations **MaxIter** and different horizon lengths  $N$ . Note that the horizon cost will differ for different horizon lengths, as an additional stage cost is included for each increment to the horizon length. Very similar trajectories are pursued for different horizon lengths, suggesting that, given enough iterations, extending the horizon length does not seem to produce superior performance, nor contribute to instability.

Fig. 4.11 shows the simulation trajectories for a fixed horizon length,  $N$ , and different maximum allowed number of iterations, **MaxIter**. Again, the controllers pursue similar trajectories for different values of **MaxIter**, suggesting that the real-time performance of the MPC does not appear to improve for marginally greater numbers of iterations, i.e. the iterates converge quite quickly to close-to-optimal values and so do not require additional permitted iterations.



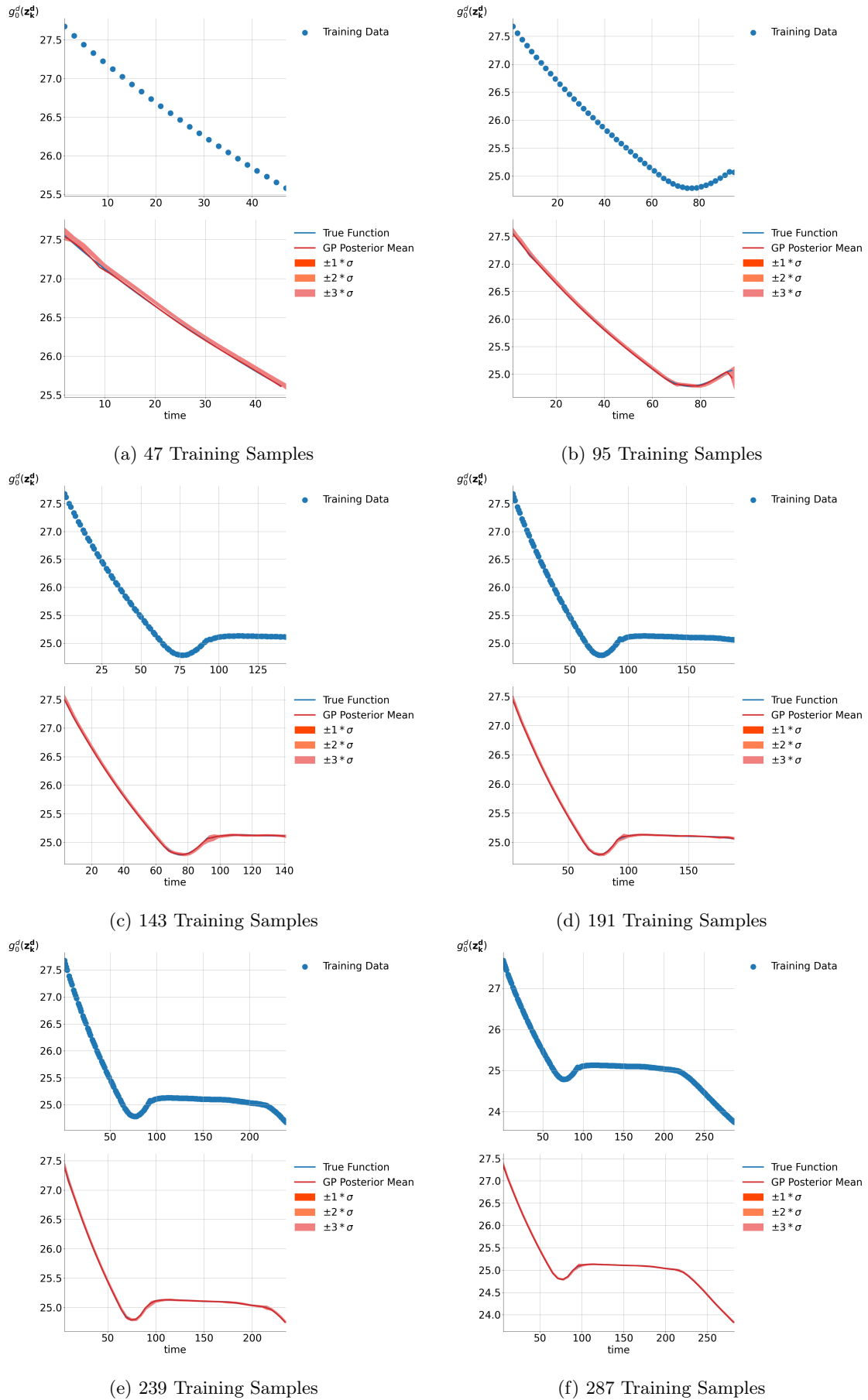


Figure 4.3: GP Predicted Output vs. Time for  $T_{k+1} = g(T_k, P_k, T_{o,k})$ . The upper plots show the training data used, whereas the lower plots show the test points (pulled from the real data at alternating indices) at which the predictions are made.

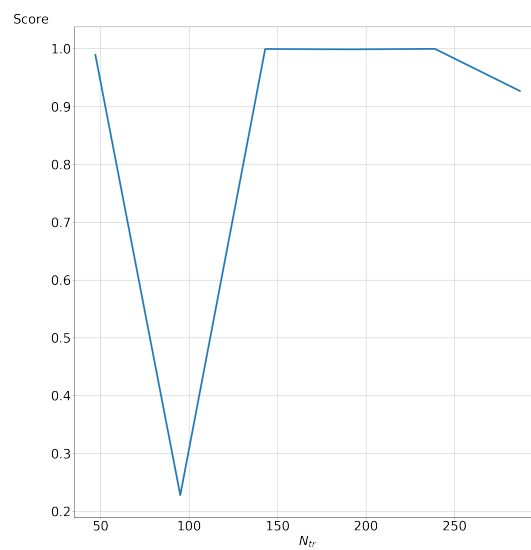


Figure 4.4: State GP Prediction Score vs. No. Training Samples for  $T_{k+1} = g(T_k, P_k, T_{o,k})$ , where predictions are made for the hour following the training data.

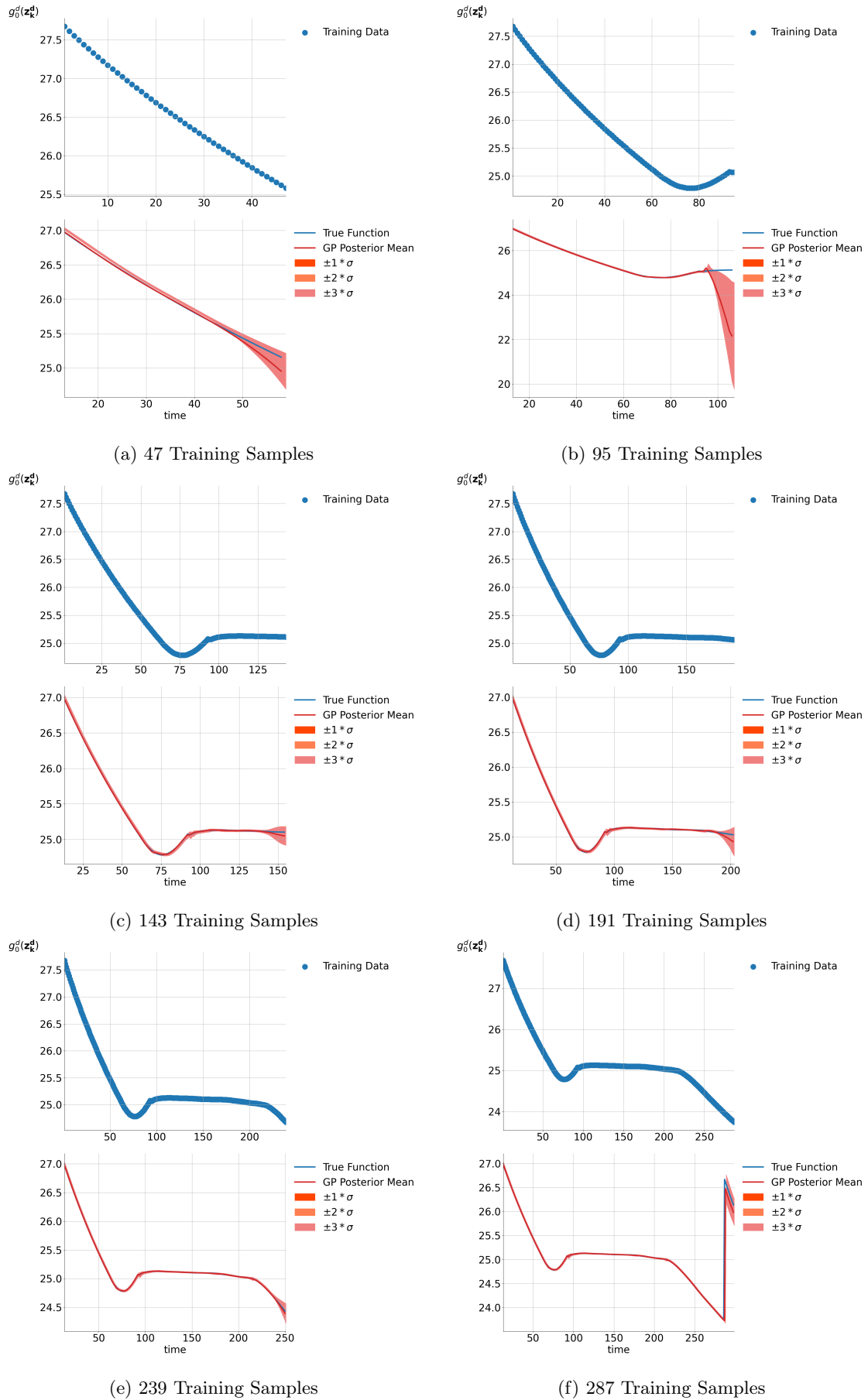


Figure 4.5: GP Predicted Output vs. Time for  $T_{k+1} = g(T_k, P_k, T_{o,k})$ . The upper plots show the training data used, whereas the lower plots show the test points (pulled from 1 hour of the real data after the selected training data) at which the predictions are made.

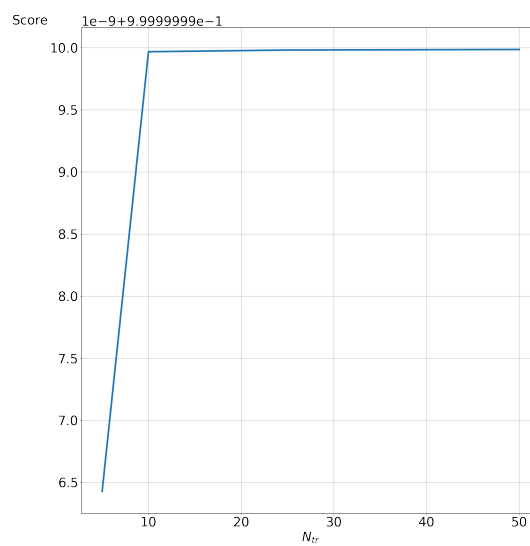


Figure 4.6: Stage Cost GP Prediction Score vs. No. Training Samples for  $l^{TCL} = j(T_k, P_k, T_{o,k})$  (Note that in the scale along the y-axis, 10.0 corresponds to 1)

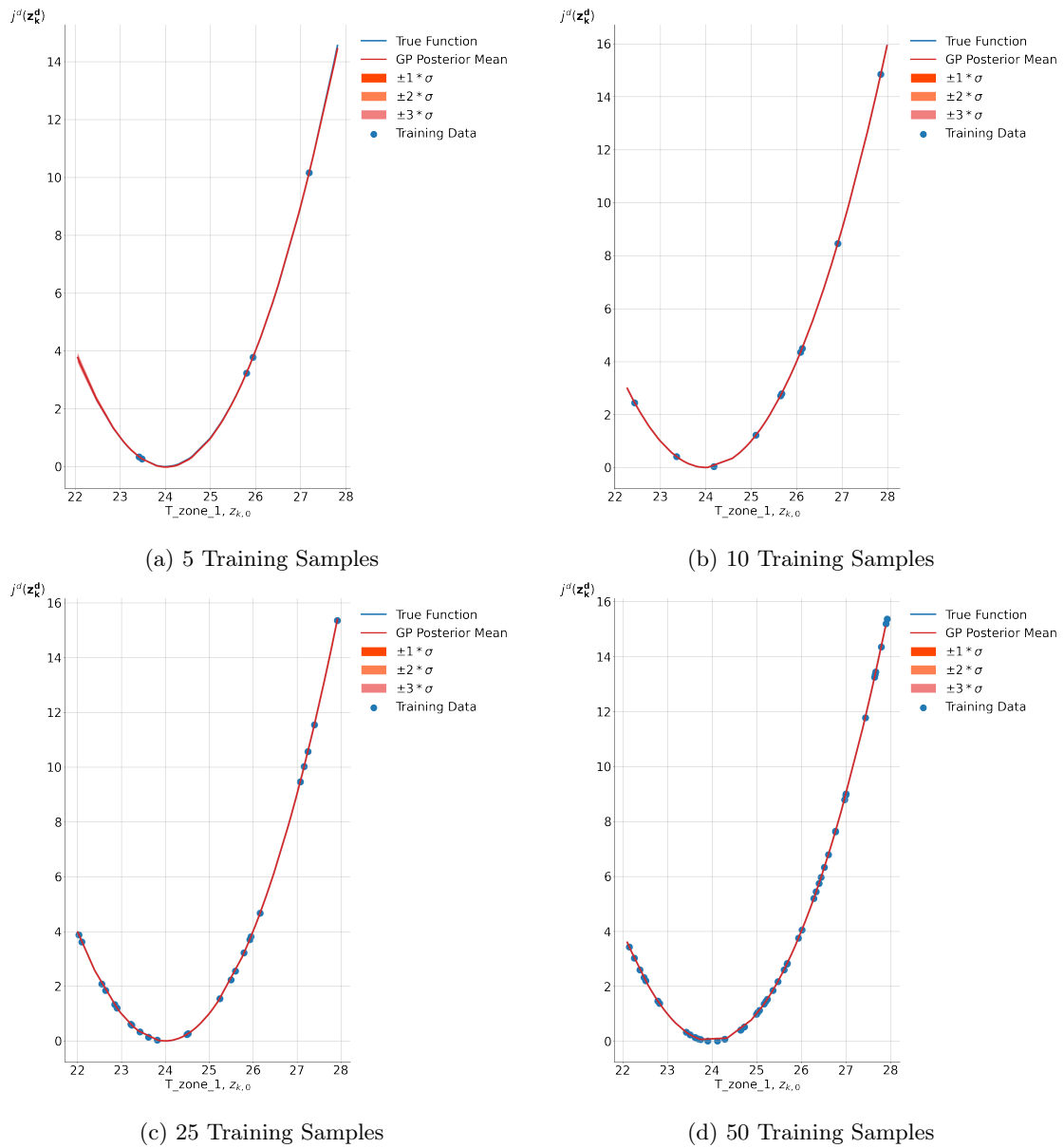


Figure 4.7: Stage Cost GP Predicted Output vs. Input Feature  $T_{zone}$  for  $l_k = j(T_k, P_k, T_{o,k})$ .

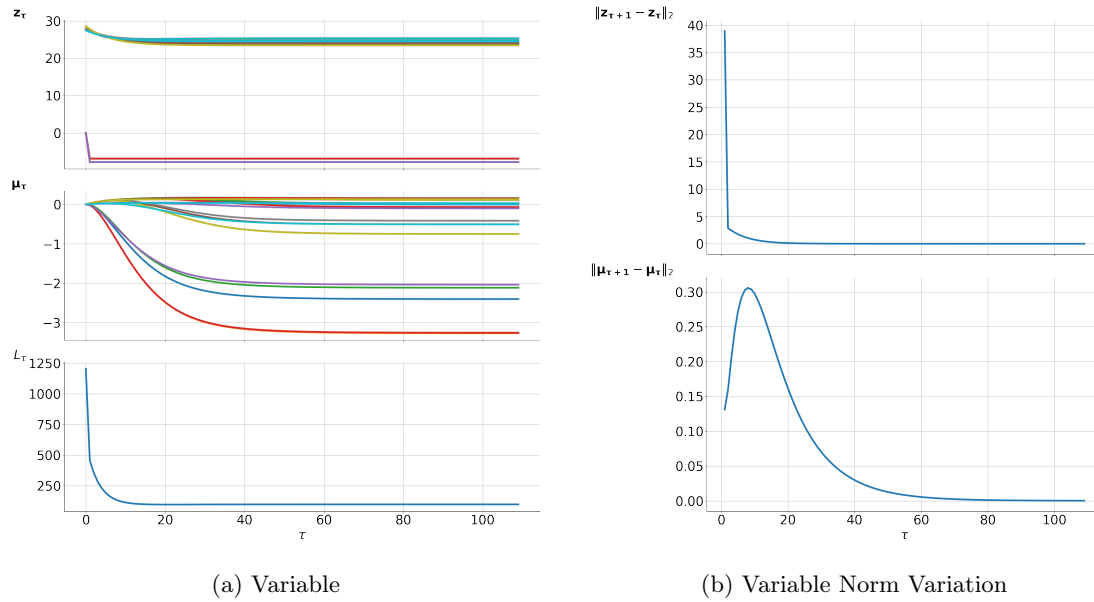


Figure 4.8: Batch Convergence for  $k_0 = 0$ ,  $N = 6$ ,  $\text{MaxIter} = 2000$

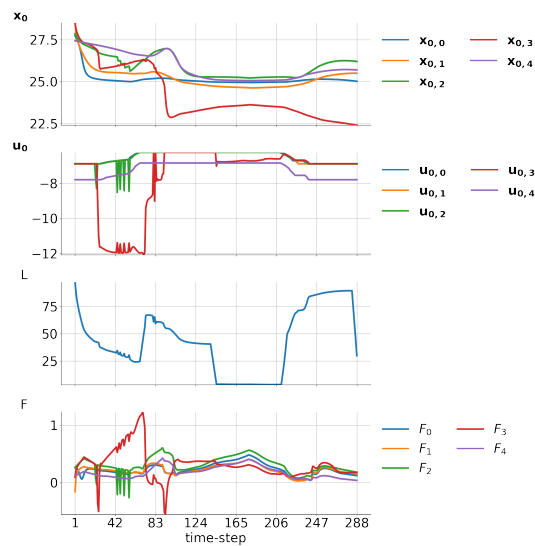
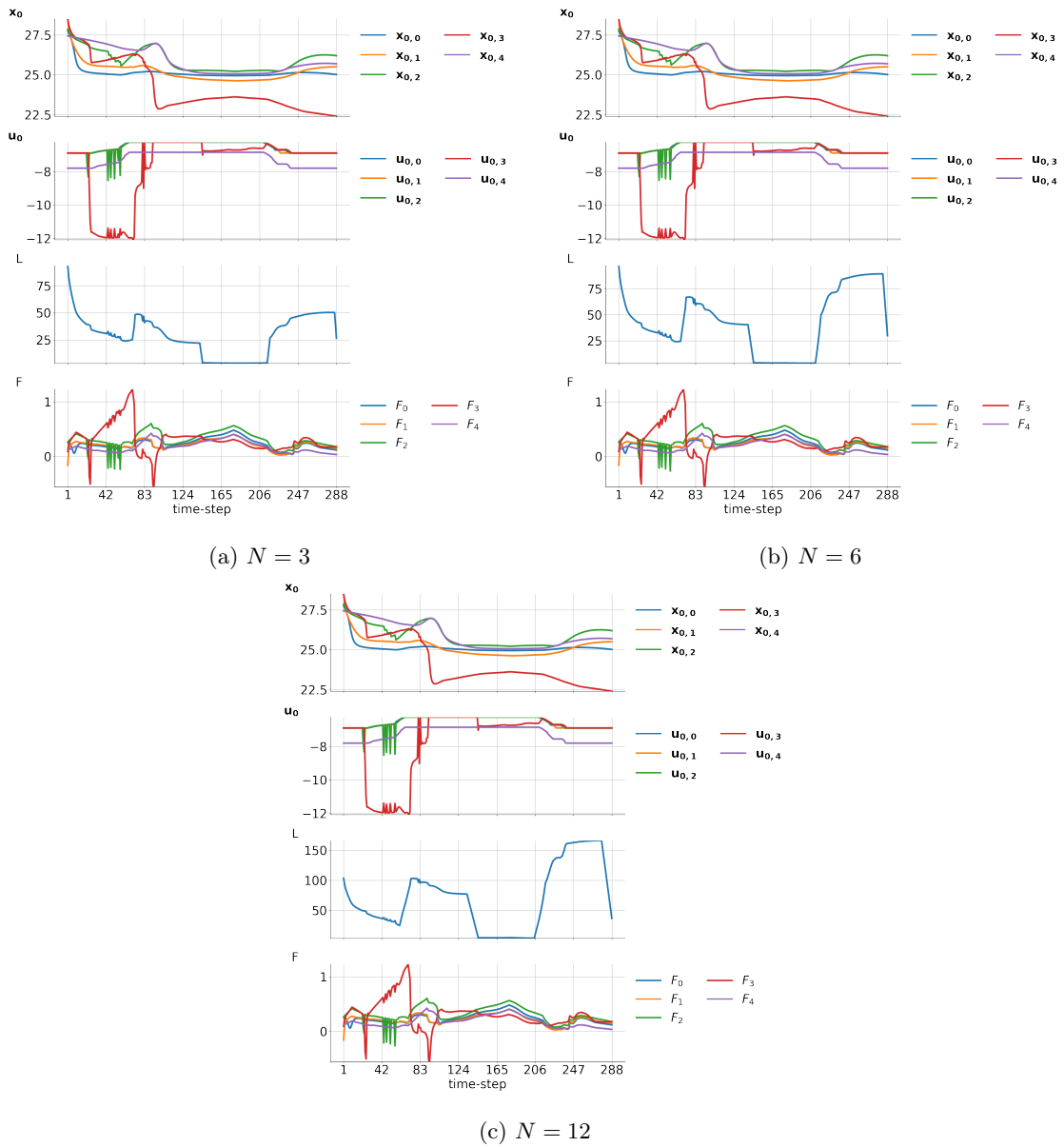


Figure 4.9: Batch MPC with GP Simulation Trajectories for  $N = 6$ ,  $\text{MaxIter} = 2000$

Figure 4.10: Real-Time Simulations for Different Horizon Lengths, `MaxIter` = 100

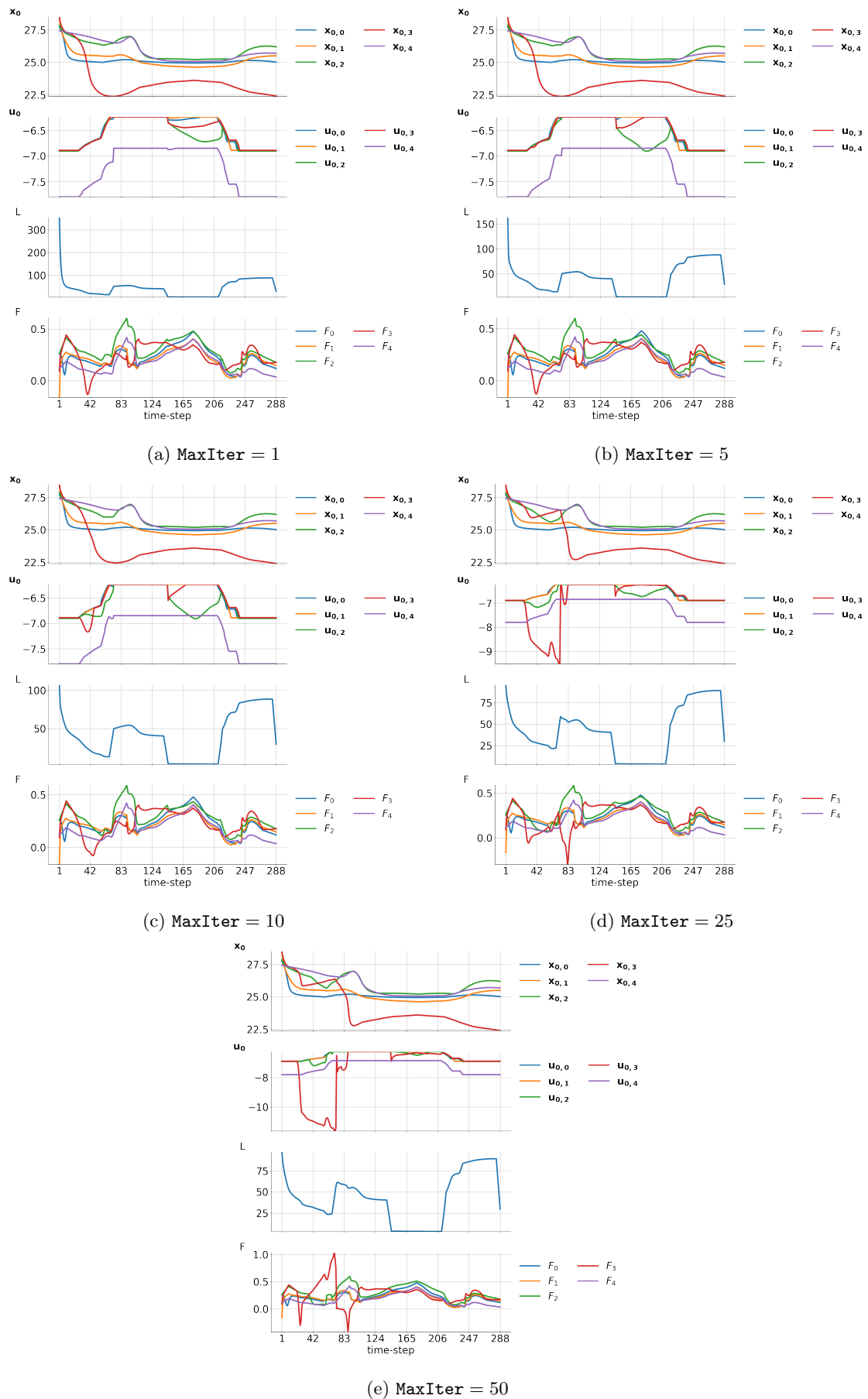


Figure 4.11: Real-Time MPC with GP Simulation Trajectories for Different MaxIter,  $N = 6$





# Chapter 5

## Conclusions

In this chapter, we will reflect upon the results of this work and their contribution to the challenges outlined in Chapter [1](#).

### 5.1 Gaussian Process-Learned Functions

Modeling nonlinear black-box functions with Gaussian Process regression has both benefits and limitations. Although GPs are in the category of nonparametric modeling approaches, the structure of the covariance kernel must be selected and the hyperparameters of these kernels must be chosen appropriately. The squared exponential covariance kernel is a reliable choice for smooth functions and is well-established in GP applications in literature, and the hyperparameters can be optimized over by maximizing the marginal log likelihood. For particular function classes (e.g. functions with quadratic and sinusoidal terms, or functions with a predictable frequency of fluctuations along each dimension of its input vector), very few training samples are necessary to model the function to a high degree of accuracy. The low number of samples required is of particular relevance when we consider systems which are expensive or burdensome to test extensively. Human users, for example, do not generally wish to be queried for feedback 2000 times at 1 second intervals. The second advantage of a low sampling rate applies to the real-time setting of the MPC scheme proposed. If the MPC relies on new information regarding the system to best optimize for a control input that is required at short, regular intervals, we cannot afford to invert a large covariance matrix each time new training data is received. The computational burden of using a GP-learned function can be minimized with adaptive sampling, by employing a low sampling rate, by implementing more efficient matrix inversion techniques and by using a small but relevant training set (i.e. considering only relevant states, inputs and/or disturbances). It should also be noted that the target of the GP can be more precisely specified if it is used exclusively for the unknown nonlinear terms of the function. This can be achieved with the use of a prior function that equates to the known terms of the function or to an established synthetic model. The additional advantage of this approach, is that if the GP is tested in a relatively unexplored region where the posterior mean may be close to 0, the value returned could depend on the synthetic model or known linear terms instead of solely on the GP prediction.

If the GP model is tested for inputs which deviate greatly from the available training data (as measured by the degree of covariance between the test inputs and the training inputs), then it is likely to return its mean prior (usually 0) with a high predicted standard deviation. High-values of the standard deviation can thus be considered as a warning signal that the GP is not reliable. In this work, the individual states and inputs of the system are constrained by the bounds of the available training data. This can be achieved by setting either dynamic bounds on the states and inputs of the system, or a dynamic terminal set which constrains the system to remain in the safely predictable region of the GP. Another approach would be to use a hybrid system function in which a synthetic model is used when states and inputs stray outside the training dataset and a GP model is used otherwise. A less conservative approach would be to improve the reliability of

the GP in real-time by executing exploratory testing of the system within the MPC to learn the unknown regions of the function. This could be achieved by incorporating the standard deviation of the predictions into the stage cost function [15] and thereby allowing the states or inputs to occasionally stray outside the safe region for the purposes of fetching useful system observations.

In order to maintain an up-to-date model of the system with a minimal number of training samples, in this work a sliding-window subset of online training data with a capped length is sampled as the control scheme is run. While this requires that the training data covariance matrix be inverted and that the hyperparameters be re-optimized more frequently, it ensures that the GP model reflects current likely disturbance values well and becomes more familiar with the current input region as it moves towards and through it. Alternatively, a running score could be calculated for the GP based on the deviation between the true and predicted system outputs, and a new set of training data collected only if the score has fallen below a given threshold.

It was noted in the inverted pendulum experiment in Chapter A that care should be taken when using a GP to estimate the rate of change of a system, as the difference between a positive and negative rate-of-change may have a significant effect on the sign of the optimal control input. This issue arose in the case of the inverted pendulum system as the rate-of-change approached 0, but the GP may have modeled a different sign from the true function, and so the optimized control input could accelerate the pendulum in the wrong direction. A solution to this problem was found by dynamically updating the online subset of the training data, such that as the trajectory approached this cross-over point, the online subset contained high-granularity data in the neighborhood of this cross-over region. While this worked well for the batch approach, it was not sufficient to stabilize the system in the online approach.

## 5.2 MPC Schemes with GP-Learned Functions

GP-learned functions are highly applicable to time-varying problems, in which the cost function and/or dynamic state function rely on multiple variables, and the synthetic model cannot capture the finer dependencies on both system and/or environmental variables. Furthermore, as the system evolves over time, GPs can capture these changes by updating their training sets. GP-models are particularly conducive to modeling systems with a low and/or heterogeneous feedback sampling rate i.e. they can generate a sufficiently accurate model with very few dissimilar, noisy training points. The human-in-the-loop scenario, in which feedback on human preferences is required to learn about their needs, is an excellent example of this, as we cannot expect users to provide consistent and frequent feedback. The network of TCLs in a building is another good example of these ‘low feedback rate’ systems, in which case it is costly to take the building offline for the purposes of performing regular, thorough testing, but feedback can be collected during normal operation in an online fashion.

A challenge presented by the MPC with GPs scheme presented in this work is the parameterization required. While the GPs corresponding to individual devices and users can be optimized over using the maximum marginal log likelihood method described in Sec. 2.5, it is less straightforward to optimize over the parameters of the regularized primal-dual gradient algorithm  $\alpha$ ,  $\eta$  and  $\epsilon$ . We found that if a set of step-sizes performs well for the batch case, then it can be expected to work for the online implementation. The same cannot be said, however, for MPC using the true functions compared to using the GP-learned models (tested in the inverted pendulum experiments in Appendix A). In this case, the step-sizes usually have to be adapted in response to the different Lipschitz constants of the true vs. GP-modeled functions. It was noted that, in general, it is safer to underestimate step-sizes than to overestimate, or to reduce them as the problem converges. Additionally, if the convergence trajectories of specific primal or dual variables are proving to be problematic, it is advisable to fine-tune the step-size associated with those particular variables.

### 5.3 Recommendations for Further Work

There are a number of modifications which could be integrated into the learning-based MPC methodology proposed and tested in this work as well as potential for further investigation into safety and/or performance guarantees for such a control scheme. In this section, we summarize some areas of high potential for research that could build on the developed framework and findings of this work:

- The real-time control scheme presented in this work could further be adapted to a distributed setting, where we consider the cost of time delays in transmitting information between devices and a central controller by leveraging the decomposability of the regularized Lagrangian function [3,20].
- A *zeroth-order* scheme could be integrated into the real-time MPC framework which leverages point-wise values of the cost function (known as *partial-information feedback*) rather than directly computing the gradient (known as *full-information feedback*), as in [30], when closed-form expressions for the derivative of the function are not available, e.g. when implementing alternative nondifferentiable covariance kernels or employing hybrid synthetic-GP models.
- A terminal cost and ‘safe’ terminal set, such as the elliptical terminal set derived and implemented in [23], could be integrated to ensure that the MPC trajectory steers the system to a region which robustly satisfies constraints with a high probability, given the propagated uncertainty of the GP predictions.
- A term maximizing the standard deviation of the GP predictions could be included in the stage cost, i.e. the Upper-Confidence Bounds (GP-UCB) method implemented in [23], such that the controller simultaneously explores unknown regions of the dynamic state GPs while exploiting the optimizer of known regions.
- Auto-regressive inputs could be considered in the GP models. This could be particularly applicable to the user dissatisfaction cost functions, where a user’s perceived dissatisfaction is likely not only to depend on the current states, inputs and disturbances, but also on historic ones.
- The building-control simulation could be augmented with uncontrolled loads, two-way loads which can produce as well as consume power (e.g. EV batteries) and convex inequality constraints limiting the power flow through individual feeder lines connecting the devices.
- In the inverted pendulum experiment in Chapter A, MPC simulations were compared for true vs. GP-modeled functions. It would be beneficial to formally characterize the difference in stability and performance for a control scheme based on true stage cost and dynamic state functions vs. GP-modeled counterparts.
- The sparsity of the covariance matrix could be leveraged to more efficiently invert it for new training data additions made online, as proposed in [17].
- Optimal Experiment Design (OED) techniques could be adapted for online learning, i.e. to determine which of the incoming feedback samples to add to the training set, and which existing samples to discard to free memory for them.



# Appendix A

## Additional Application to an Inverted Pendulum System

Chapter 4 illustrated the merits of the proposed methodology in the context of energy systems. In this chapter, we provide an additional illustrative example in which the proposed MPC with GPs algorithm is used to solve an inverted pendulum system (similar to the case study conducted in 23) in real-time. All Python code written for the experiments conducted in this chapter can be found at 51.

### A.1 System Model

In this section, we will describe the states, inputs, disturbances and parameters which characterize the inverted pendulum system.

#### States

The states of the inverted pendulum system consist of the angle  $\theta$  of the pendulum from the upright position and the angular velocity  $\dot{\theta}$  of the pendulum's rotation, see A.1.

$$\mathbf{x}_k := \begin{bmatrix} \theta_k [\text{rad}] \\ \dot{\theta}_k [\text{rad/s}] \end{bmatrix} \quad (\text{A.1a})$$

$$\mathcal{X}_k := \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \begin{bmatrix} \underline{\theta} \\ \underline{\dot{\theta}} \end{bmatrix} \leq \mathbf{x} \leq \begin{bmatrix} \bar{\theta} \\ \bar{\dot{\theta}} \end{bmatrix} \right\} \quad \forall k = 0, 1, N-1 \quad (\text{A.1b})$$

$$\mathcal{X}_f := \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \begin{bmatrix} \underline{\theta} \\ \underline{\dot{\theta}} \end{bmatrix} \leq \mathbf{x} \leq \begin{bmatrix} \bar{\theta} \\ \bar{\dot{\theta}} \end{bmatrix} \right\} \quad (\text{A.1c})$$

#### Control Inputs

The control inputs of the system consist of the applied torque  $\tau^\theta$ , see A.2.

$$\mathbf{u}_k = [\tau_k^\theta [Nm]] \quad \forall k = 0, 1, \dots, N \quad (\text{A.2a})$$

$$\mathcal{U} = \{u \in \mathbb{R} \mid \underline{\tau}^\theta \leq u \leq \bar{\tau}^\theta\} \quad (\text{A.2b})$$

#### Disturbances

The disturbances of the system consist of a disturbance torque  $\tau_w^\theta$ .

$$\mathbf{w}_k = [\tau_{w,k}^\theta [Nm]] \quad \forall k = 0, 1, \dots, N-1 \quad (\text{A.3a})$$

### Dynamic State Equation

The true continuous-time dynamics of the system are given by Eqn. [A.4](#).

$$\dot{\mathbf{x}} := f(\mathbf{x}, \mathbf{u}; \mathbf{w}) = \left[ \frac{g}{l} \sin \theta - \frac{\eta}{ml^2} \dot{\theta} + \frac{1}{ml^2} (\tau^\theta + \tau_w^\theta) \right] \quad (\text{A.4})$$

where:

$m$  is the mass of the pendulum [kg]

$l$  is the length of the pendulum [m]

$\eta_f$  is the friction parameter [Nms/rad]

$g$  is the gravitational constant [m/s<sup>2</sup>]

This continuous set of equations is discretized with the Runge-Kutte fourth-order method (see Appendix [D.2.2](#)). The derived difference equations are lengthy and are thus omitted.

### Stage Cost

The true stage cost and terminal cost are given by [A.5a](#) and [A.5b](#), respectively.

$$l(\mathbf{x}_k, \mathbf{u}_k) = \theta_k^2 \quad (\text{A.5a})$$

$$l_f(\mathbf{x}_N) = \theta_N^2 \quad (\text{A.5b})$$

### Parameters

The parameters of the system include the parameterization of the dynamic state function [A.4](#) and the upper and lower bounds of the angle  $\theta$ , the angular velocity  $\dot{\theta}$  and the applied torque  $\tau^\theta$ , see [A.6](#).

$$\mathbf{p}_k := \left[ m, g, l, \eta, \bar{\theta}_k, \underline{\theta}_k, \bar{\dot{\theta}}_k, \underline{\dot{\theta}}_k, \bar{\tau}^\theta_k, \underline{\tau}^\theta_k \right]^T \quad (\text{A.6})$$

## A.2 Results of GP Predictions

In this section, we provide plots of the GP predictions for both the nonlinear dynamic state equation and stage cost and scores of the GP models.

### A.2.1 State Variation GP Predictions

In the case of the inverted pendulum system, we only model the dynamic equation of the second state,  $\dot{x}_2 = \ddot{\theta}$  with a GP. The reason for this is that we know that the first state is a linear function of the inputs,  $\dot{x}_1 = x_2$ , and so there is no need to employ a nonlinear nonparametric regression technique to learn this. The continuous equations [A.4](#) are first discretized using the Runge-Kutte fourth-order method (see Appendix [D.2.2](#)), such that the discrete-time next state  $\dot{\theta}_{k+1}$  can be estimated with a GP. The only nonlinear component in the discretization of Eqn. [A.4](#) is the sine term, which is a function of  $\theta$ . We therefore only plot the GP model of  $\dot{\theta}_{k+1}$  as a function of  $\theta$ . Although linear terms can be very precisely approximated by GPs and are trivial to parameterize (we can simply set the length scale to a number arbitrarily greater than the expected input domain), there are many less computationally intensive parametric methods of approximating linear models e.g. linear regression. While we do not do this for the plots shown for consistency with the first

experiment (see [4](#)), if the linear terms of an unknown function are known, we can set the prior  $h$  to equal these terms, such that they are subtracted from the training data and the GP does not have to consider their influence.

Table [A.1](#) shows the initial hyperparameters and scores for different numbers of training samples. The hyperparameters are still found by maximizing the marginal log likelihood, but the initial values can significantly influence the solution if there are multiple local optima. In this case we know that the variation of the nonlinear sinusoidal term can be modeled with a length scale of  $\pi$  and the variation of the linear terms of the two remaining variables  $\dot{\theta}$  and  $\tau^{\theta}$  can be modeled with arbitrarily large length scales. Fig. [A.1](#) shows the score of the GP and Fig. [A.2](#) shows the evolution of the GP prediction and standard deviation with increasing numbers of training samples. We can see, that with only 10 training samples the variations in the sinusoidal relationship are very precisely captured.

$N_{tr}$	$N_*$	$\mathbf{l}_g$	$\sigma_g^2$	$\sigma_{n,g}^2$	Score
5					0.7048
10	50	[3.1416, 100, 100, 100]	10.0	$10^{-6}$	1.0
25					1.0
50					1.0

Table A.1: State GP Predictions for  $x_2 = \dot{\theta}$

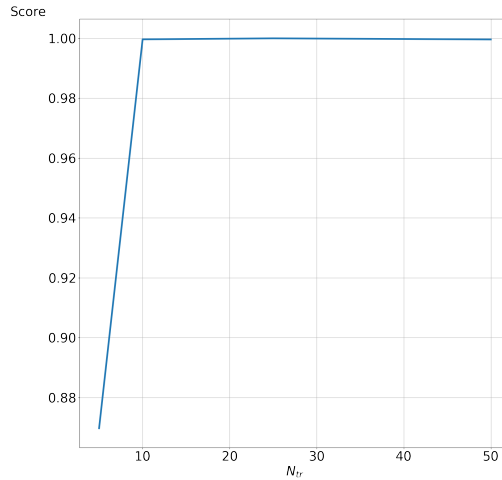


Figure A.1: State GP Prediction Score vs. No. Training Samples for  $x_2 = \dot{\theta}$

It appears that univariate sinusoidal functions are straightforward to parameterize based on intuition. The challenge, is that in real applications, we will know very little, if anything about the structure of the true function. However, if its synthetic equivalent is approximated by a quadratic or sinusoidal function, we can at least set approximate initial values to the hyperparameters, from which they can be further optimized by maximizing the marginal log likelihood function (see Sec. [2.5](#)).

## A.2.2 Cost GP Prediction

The stage cost was synthesized as the squared deviation of the angle  $x_1 = \theta$  from the upright position  $\theta = 0$ . One challenge that arises with this formulation, is that if the pendulum swings around  $360^\circ$ , this cost function evaluates to  $(2\pi)^2$  rather than 0. Additionally, a pendulum that



swings beyond the downward position by  $1^\circ$  results in a higher cost function than if it stops  $1^\circ$  short of this mark. A more accurate model would reflect the symmetry of the problem and could be formulated as  $\min(\text{mod}(\theta/2\pi), 2\pi - \text{mod}(\theta/2\pi))$ . While this was tested, it resulted in a more complex, periodic cost function and so for the sake of simplicity the basic quadratic function is used for the experiments conducted in this work, while the variables  $\theta$  and  $\dot{\theta}$  are instead constrained to be between  $-\pi$  and  $\pi$ .

Table A.2 gives the parameterization of the stage cost function with the corresponding scores for increasing numbers of training samples. Note how, since the function is known to be independent of variables  $\tau^\theta$ ,  $\tau_w^\theta$  and  $\theta$ , we can safely set the corresponding length scales to arbitrarily small values, thereby implementing Automatic Relevance Determination. Fig. A.3 shows the increase in GP score and Fig. A.4 illustrates the GP model of the quadratic stage cost for increasing numbers of training samples. As can be expected for a quadratic function parameterized based on its domain, the GP has attained a near perfect model with only 5 training samples.

$N_{tr}$	$N_*$	$\mathbf{l}_1$	$\sigma_l^2$	$\sigma_{n,l}^2$	Score
5					1.0
10	50	$[62.8319, 10^{-6}, 10^{-6}, 10^{-6}]$	50	$10^{-6}$	1.0
25					1.0
50					1.0

Table A.2: Cost GP Predictions

### A.3 Results of MPC Simulations

In this section the results of the MPC simulation for a single inverted pendulum over 300 time-steps are compared for the true functions and the GP-modeled functions. Refer to Sec. 4.2 for a description of the variables plotted. The inverted pendulum and GP parameters are given in Tables A.3 and A.4, respectively.

$m$	$g$	$l$	$\eta$	$\underline{\theta}_k$	$\bar{\theta}_k$	$\dot{\underline{\theta}}_k$	$\dot{\bar{\theta}}_k$	$\underline{\tau}^\theta$	$\bar{\tau}^\theta$
0.15	9.81	0.5	0.1	$-\pi$	$\pi$	$-\pi$	$\pi$	$-1$	1

Table A.3: Inverted Pendulum Parameters

Dynamic State/Stage Cost Function	$N_{tr,0}$	$N_{tr}$	$\Delta t_x^d / \Delta t_l^d$ [s]
Dynamic State	100	200	0.1
Stage Cost	25	50	0.1

Table A.4: GP Parameters

The disturbances were set as:

$$\tau_{w,k_0}^\theta = \begin{cases} -0.5 & \text{for } k_0 \in [74, 76], [224, 226] \\ 0.5 & \text{for } k_0 \in [149, 151] \\ 0 & \text{otherwise} \end{cases}$$

Fig. A.5 shows the convergence of the primal variables (top plot), dual variables (middle plot) and horizon cost (bottom plot) and Fig. A.6 shows the convergence of the L2-norm of the change between adjacent iterations for the first simulation time-step  $k_0 = 0$ . While there are some

Function Models	$\Delta t$	MaxIter	$\nu$	$T$	$N$	$\alpha$	$\eta$	$\epsilon$
True Model	0.1	2000	$10^{-4}$	300	3	0.03	1	1
GP Model								

Table A.5: Batch MPC Simulation Parameters

oscillations in the convergence, the variables clearly settle to an optimum value - although several hundred iterations are required.

Fig. [A.7](#) shows the trajectory of the true states  $\mathbf{x}_0$  (fetched from the true system [A.4](#)), optimized control inputs  $\mathbf{u}_0$ , horizon cost  $L(\mathbf{z})$  and equality constraint violation  $F(\mathbf{z})$  over a simulation period of 300 0.1-second time-steps (see Sec. [4.2](#) for a full description of the variables). The trajectories for the batch approach are compared for the case when the true functions are used in Fig. [A.7a](#) and when the GP-learned functions are employed instead in Fig. [A.7b](#). Clearly, the trajectories are very similar, with small variations around the true optima when GPs are used to model the system and stage cost.

While online experiments were conducted for the inverted pendulum system, it was found to perform poorly for the true functions across a range of maximum iterations and horizon lengths and for the GP-modeled functions to become unstable. This could at least in part be due to the nature of the control logic: when the time derivative of the angle ( $x_2 = \dot{\theta}$ ) is equal to 0, the pendulum is seen to be stationary. When it is positive (negative), the pendulum is seen to be accelerating in the clockwise (counter-clockwise) direction. The optimal control input is thus highly sensitive to the distinction between negative, null and positive values of the second state. The result, is that inaccuracies, even small ones, in the GP model around the 0 point make it difficult to control the system with very few iterations.

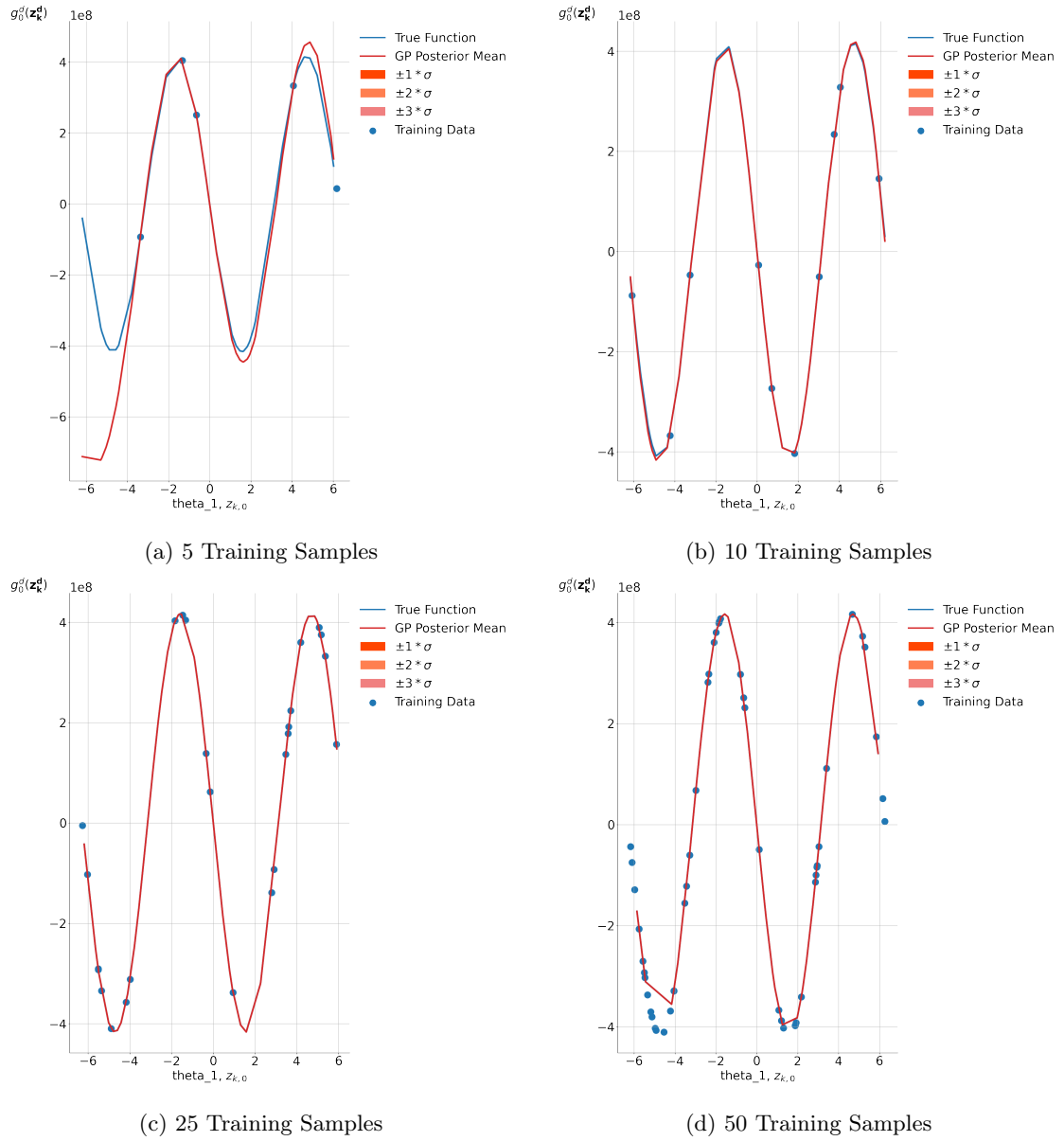


Figure A.2: State GP Predicted Output vs.  $\theta$  for  $\dot{\theta}_{k+1} = g(\theta_k, \hat{\theta}_k, \tau_k^\theta, \tau_{w,k}^\theta)$

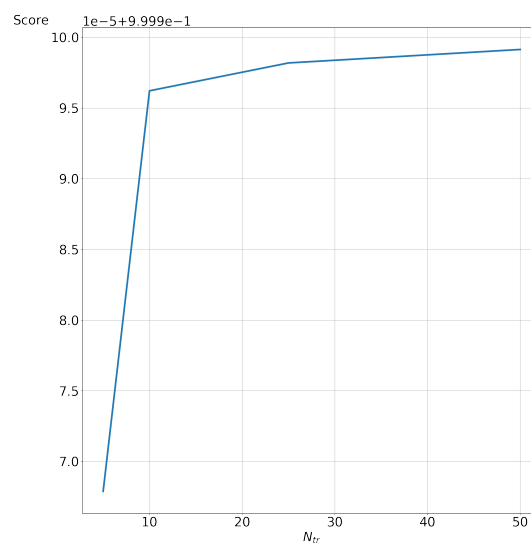
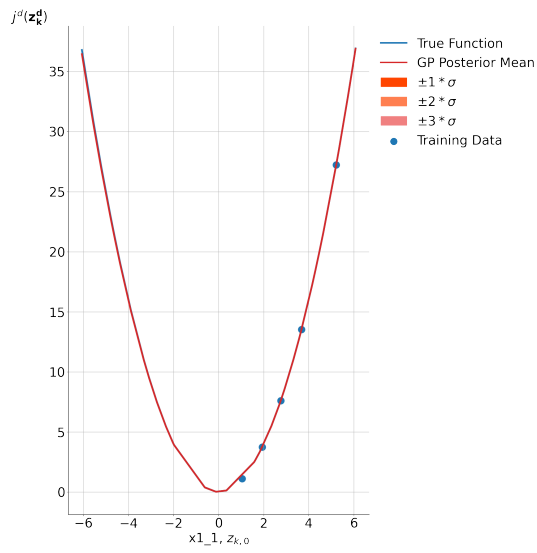
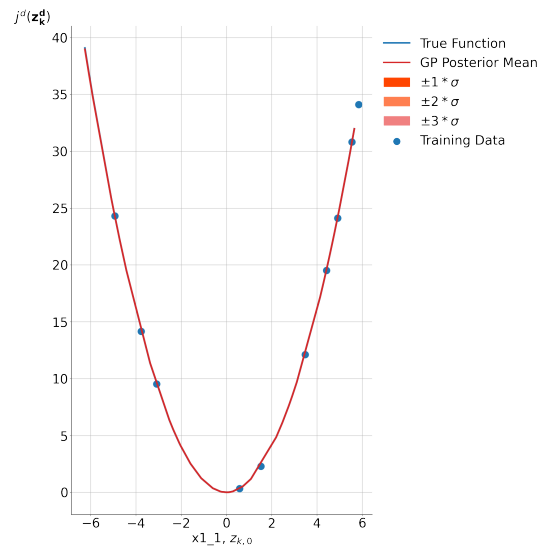


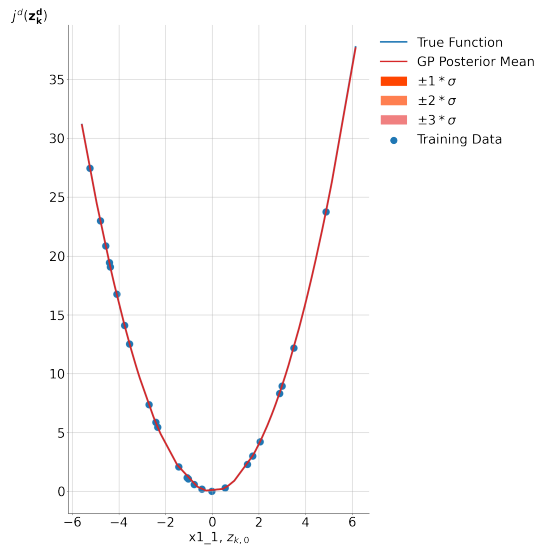
Figure A.3: Stage Cost GP Prediction Score vs. No. Training Samples for  $l_k = \theta_k^2$ . Note that on this y-scale, the value 10.0 corresponds to a score of 1.0.



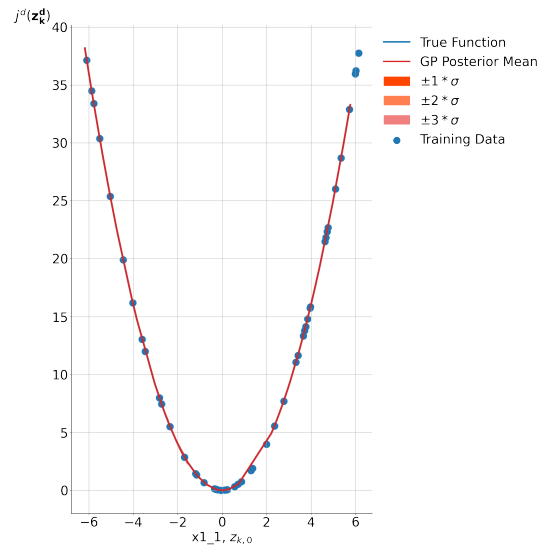
(a) 5 Training Samples



(b) 10 Training Samples



(c) 25 Training Samples



(d) 50 Training Samples

Figure A.4: Stage Cost GP Predictions for  $l_k = \theta_k^2$

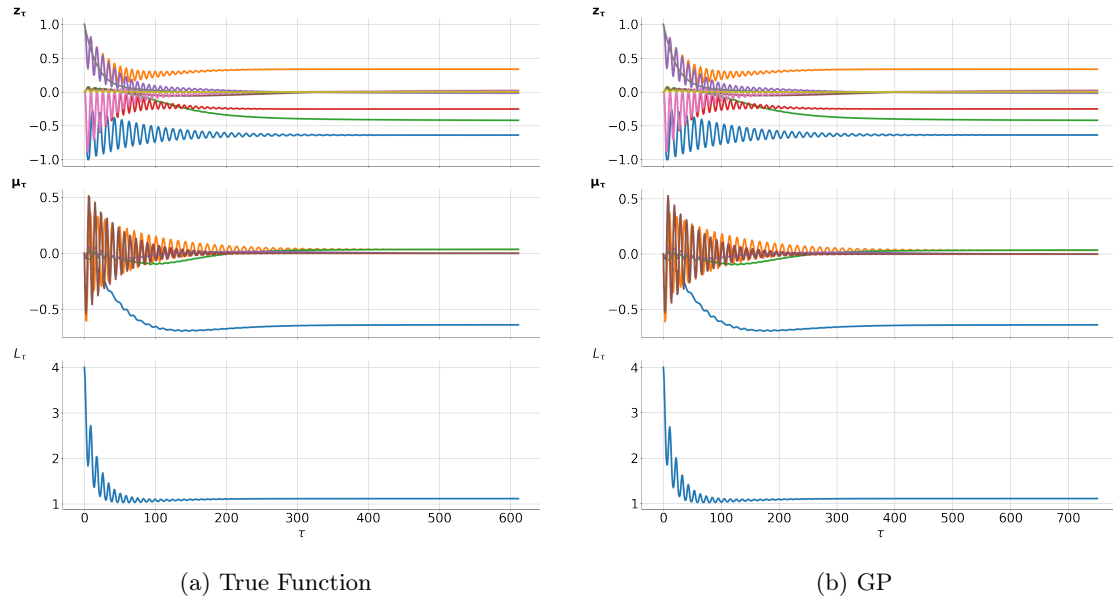


Figure A.5: Batch Variable Convergence for  $k_0 = 0$

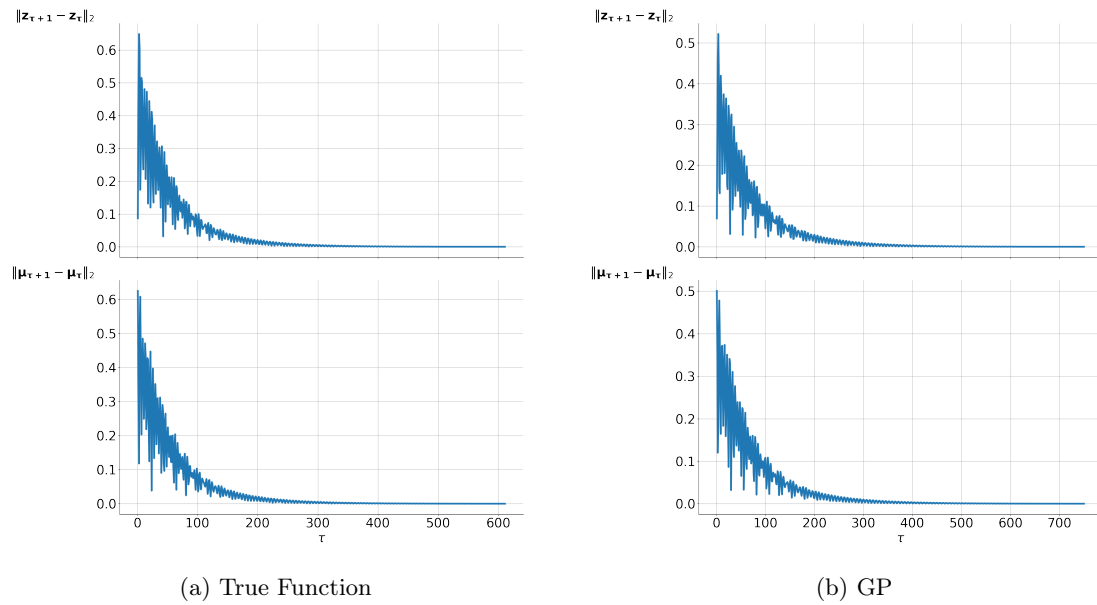


Figure A.6: Batch Variable Norm Variation Convergence for  $k_0 = 0$

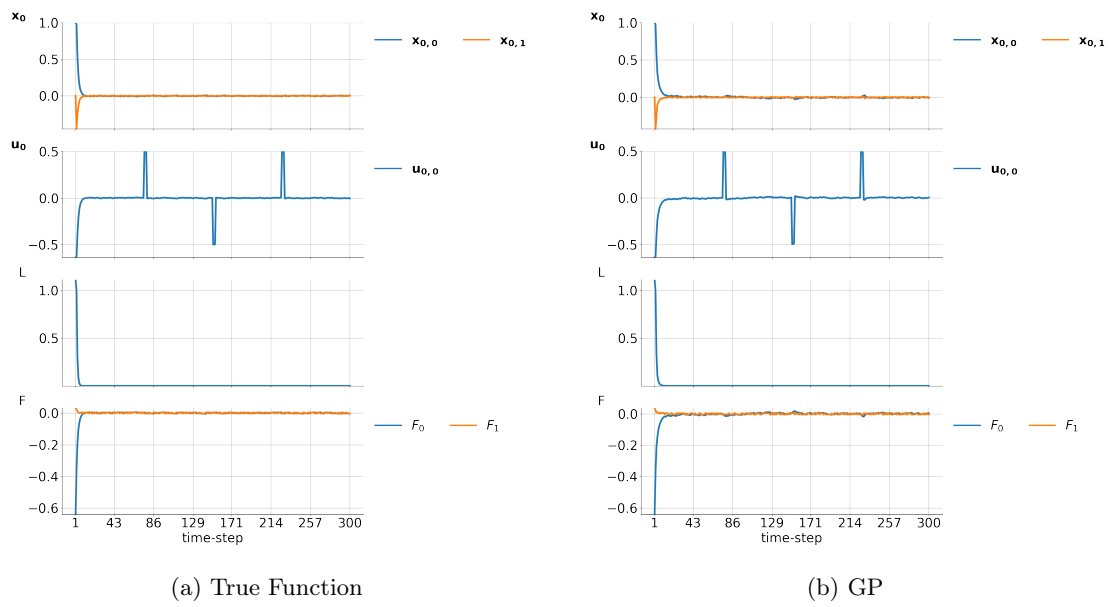


Figure A.7: Batch Simulation Trajectories

# Appendix B

## Mathematical Background

### B.1 Probability & Probability Density

This section is based on [38] Sec A.1. Let  $y$  be a (continuous or discrete) random variable. Let  $p(y)$  be the *probability density*, where  $p(y) \geq 0$  and  $\int_{-\infty}^{\infty} p(y)\partial y = 1$ , and  $\partial y$  be the *interval length*. The *probability* of sampling a given value  $y_* \in [y, y + \partial y]$  is then  $p(y)\partial y$  and a given value  $y_* \in [a, b]$  is  $\int_a^b p y \partial y$ .

Let  $\mathbf{y} = [y_1, y_2, \dots, y_n]$  be a vector of  $n$  (continuous or discrete) random variables, with a probability density function (defined over an infinitesimal interval for continuous variables) or probability function (defined over a finite interval for discrete variables) of  $p(\mathbf{y})$ . Then  $p(\mathbf{y})$  is a *joint probability* over multiple random variables.

### B.2 Joint, Marginal & Conditional Probability

This section is based on [38] Sec A.1.

Let us partition the vector of random variables  $\mathbf{y}$  into two disjoint sets:  $\mathbf{y}_A$  and  $\mathbf{y}_B$ , such that their union returns the original vector  $\mathbf{y} = \mathbf{y}_A \cup \mathbf{y}_B$  and such that their *joint probability* is expressed as  $p(\mathbf{y}) = p(\mathbf{y}_A, \mathbf{y}_B)$ .

In order to find the probability of a given vector of one or more random variables  $\mathbf{y}_A$ , we *marginalize* the joint probability over the remaining free variables  $\mathbf{y}_B$ . The *marginal probability* of  $\mathbf{y}_A$  is given by Eqn. [B.1] for continuous [B.1a] and discrete [B.1b] random variables.

$$p(\mathbf{y}_A) = \int p(\mathbf{y}_A, \mathbf{y}_B) d\mathbf{y}_B \text{ for continuous random variables} \quad (\text{B.1a})$$

$$p(\mathbf{y}_A) = \sum p(\mathbf{y}_A, \mathbf{y}_B) d\mathbf{y}_B \text{ for discrete random variables} \quad (\text{B.1b})$$

Note that if the vector  $\mathbf{y}_A$  contains more than one random variable, then the marginalized probability is itself a joint probability of these variables. If the joint probability is equal to the product of the marginal probabilities i.e.  $p(\mathbf{y}_A, \mathbf{y}_B) = p(\mathbf{y}_A)p(\mathbf{y}_B)$ , we say that  $\mathbf{y}_A$  and  $\mathbf{y}_B$  are *independent* variables, otherwise they are considered *dependent* variables.

In order to find the probability of  $\mathbf{y}_A$  *given* particular values of the vector  $\mathbf{y}_B$ , we calculate the *conditional probability*, defined for  $p(\mathbf{y}_B) > 0$  (as it is not meaningful to condition a probability on an impossible event), calculated as in Eqn. [B.2]

$$p(\mathbf{y}_A | \mathbf{y}_B) = \frac{p(\mathbf{y}_A, \mathbf{y}_B)}{p(\mathbf{y}_B)} \quad (\text{B.2})$$

If  $\mathbf{y}_A$  and  $\mathbf{y}_B$  are independent, then the marginal and conditional probabilities are equal i.e.:

$$p(\mathbf{y}_A) = p(\mathbf{y}_A | \mathbf{y}_B). \quad (\text{B.3})$$



## B.3 Bayes' Theorem

This section draws from [38] Sec A.1.

Using the expressions for conditional probabilities [B.4]:

$$p(\mathbf{y}_A | \mathbf{y}_B) = \frac{p(\mathbf{y}_A, \mathbf{y}_B)}{p(\mathbf{y}_B)} \quad (\text{B.4a})$$

$$p(\mathbf{y}_B | \mathbf{y}_A) = \frac{p(\mathbf{y}_A, \mathbf{y}_B)}{p(\mathbf{y}_A)} \quad (\text{B.4b})$$

we can express the conditional probability of  $\mathbf{y}_A$  given  $\mathbf{y}_B$  independently of the joint distribution, as in Eqn. [B.5].

$$p(\mathbf{y}_A | \mathbf{y}_B) = \frac{p(\mathbf{y}_A)p(\mathbf{y}_B | \mathbf{y}_A)}{p(\mathbf{y}_B)} \quad (\text{B.5})$$

## B.4 Gaussian Identities

This section is based on [38] Sec A.2. The multivariate Gaussian (or normal) distribution of the random variables  $\mathbf{y} = [y_1, y_2, \dots, y_n]^T$  has a joint probability density function given by [B.6a], given in the compact form by [B.6b].

$$p(\mathbf{y}_A | \boldsymbol{\mu}_A, K_A) = (2\pi)^{-\frac{n}{2}} |K_A|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{y}_A - \boldsymbol{\mu}_A)^T K_A^{-1}(\mathbf{y}_A - \boldsymbol{\mu}_A)\right) \quad (\text{B.6a})$$

$$\mathbf{y}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, K_A) \quad (\text{B.6b})$$

where:

$\boldsymbol{\mu}_A \in \mathbb{R}^n$  is *mean vector* (the mean corresponding to each random variable in  $\mathbf{y}_A$ )

$K_A \in \mathbb{R}^{n \times n}$  is the symmetric positive-definite covariance matrix

Let  $\mathbf{y}_A$  and  $\mathbf{y}_B$  be jointly Gaussian random vectors with a joint distribution given by [B.7]

$$\begin{bmatrix} \mathbf{y}_A \\ \mathbf{y}_B \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix}, \begin{bmatrix} K_{AA} & K_{AB} \\ K_{BA} & K_{BB} \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix}, \begin{bmatrix} \tilde{K}_{AA} & \tilde{K}_{AB} \\ \tilde{K}_{BA} & \tilde{K}_{BB} \end{bmatrix}^{-1}\right) \quad (\text{B.7})$$

The *marginal probability* of  $\mathbf{y}_A$  is then given by [B.8]:

$$\mathbf{y}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, K_{AA}) \quad (\text{B.8})$$

and the *conditional probability* of  $\mathbf{y}_A$  given  $\mathbf{y}_B$  is given by [B.9]:

$$\mathbf{y}_A | \mathbf{y}_B \sim \mathcal{N}(\boldsymbol{\mu}_A - \tilde{K}_{AA}^{-1} \tilde{K}_{AB}(\mathbf{y}_B - \boldsymbol{\mu}_B), \tilde{K}_{AA}^{-1}) \quad (\text{B.9})$$

Note that the marginal and conditional probability distributions of Gaussian distributions are themselves Gaussian distributions.

## B.5 Convex Sets

This section is based on [48] Sec 2.1.4.

**Definition B.5.1** (Convex Set). *A set  $C$  is defined as convex if the line segment between any two points in  $C$  lies in  $C$  i.e. if for any  $\mathbf{x}_1, \mathbf{x}_2 \in C$  and any  $\theta \in [0, 1]$ , we have:*

$$\theta \mathbf{x}_1 + (1 - \theta) \mathbf{x}_2 \in C \quad (\text{B.10})$$

*Let a convex combination of a set of points  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$  be a point of the form  $\sum_{i=1}^k \theta_i \mathbf{x}_i$ , where  $\sum_{i=1}^k \theta_i = 1$ . A set is convex if and only if it contains every convex combination of its points.*

## B.6 Compact Sets

*Compactness* is a property of topological spaces. In Euclidean space, compactness is equivalent to a set being closed and bounded. The formal definition is given by Def. [B.6.1].

**Definition B.6.1** (Compact Set). *Define an open cover of a subset  $\mathcal{S}$  of an open space  $\mathcal{X}$  as a collection of open sets that cover  $\mathcal{S}$ . We define  $\mathcal{S}$  as compact if for every open cover of  $\mathcal{S}$  there exists a finite subcover i.e. a finite number of open sets selected from each such collection can also cover the set.*

## B.7 Mean Square Continuity & Differentiability

This section draws from [38] Sec. 4.1.1.

*Mean square continuity* and *mean square differentiability* are properties of stochastic processes and help us to determine the smoothness of a Gaussian Process  $f(\mathbf{x})$  specified by a given covariance kernel  $k$ .

Let  $\mathbf{x}_1, \mathbf{x}_2, \dots$  be a sequence of points and  $\mathbf{x}_* \in \mathbb{R}^n$  be a fixed point such that the sequence approaches the fixed point i.e.  $|\mathbf{x}_k - \mathbf{x}_*| \rightarrow 0$  as  $k \rightarrow \infty$ . A GP  $f(\mathbf{x})$  is *continuous in mean square* over  $A$  if for all  $\mathbf{x}_* \in A \subseteq \mathbb{R}^n$  the mean square of the difference between the function values at the sequence point and the fixed point approaches zero i.e.  $\mathbb{E}[|f(\mathbf{x}_k) - f(\mathbf{x}_*)|^2] \rightarrow 0$  as  $k \rightarrow \infty$ .

A GP  $f(\mathbf{x})$  is continuous in mean square at the input point  $\mathbf{x}_*$  if and only if its covariance kernel  $k(\mathbf{x}_1, \mathbf{x}_2)$  is continuous for the points  $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_*$ . For a stationary kernel (a function of  $|\mathbf{x}_1 - \mathbf{x}_2|$ ) this requires that the kernel is continuous at  $\mathbf{x}_* = \mathbf{0}$ .

We say that the GP  $f(\mathbf{x})$  is differentiable in the mean square sense and has a *mean square derivative*  $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$  if Eqn. [B.11] holds along all dimensions  $i$ .

$$\lim_{h \rightarrow 0} \mathbb{E} \left[ \left( \frac{f(\mathbf{x} + h \mathbf{e}_i) - f(\mathbf{x})}{h} - \frac{\partial f(\mathbf{x})}{\partial x_i} \right)^2 \right] = 0 \quad (\text{B.11})$$

where:

$\mathbf{e}_i$  is the unit vector in the  $i^{\text{th}}$  direction

The  $k^{\text{th}}$  order mean square partial derivative of  $f(\mathbf{x})$  exists for all  $\mathbf{x} \in \mathbb{R}^{n_x}$  (i.e.  $f$  is *smooth* for all  $\mathbf{x}$ ) if the  $2k^{\text{th}}$  order partial derivative  $\frac{\partial^{2k} k(\mathbf{x})}{\partial x_i^{2k}}$  (which is the covariance function of the  $k^{\text{th}}$  derivative  $\frac{\partial f(\mathbf{x})}{\partial x_i}$ ) exists and is finite at  $x_i = 0$  for all dimensions  $i$ .



# Appendix C

## Optimization Theory

### C.1 Convex Functions

This section is based on [48] Sec. 3.1.1.

A function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  is *convex* if  $\text{dom}(f)$  is a convex set (see Appendix B.5) and if for all  $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$  and  $\theta \in [0, 1]$ , we have:

$$f(\theta\mathbf{x} + (1 - \theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1 - \theta)f(\mathbf{y}). \quad (\text{C.1})$$

We say that  $f$  is *concave* if  $-f$  is convex. All affine functions are both convex and concave.

### C.2 Convex Optimization Problems

This section is based on [48] Sec. 4.2.

A *convex optimization problem* is one of the form given by C.2.

$$\min_{\mathbf{x} \in \mathcal{D}} f_0(\mathbf{x}) \quad (\text{C.2a})$$

$$\text{s.t. } f_i(\mathbf{x}) \leq 0 \forall i = 1, \dots, m \quad (\text{C.2b})$$

$$a_i^T \mathbf{x} = b_i \forall i = 1, \dots, p \quad (\text{C.2c})$$

where:

the objective function  $f_0$  is convex

the inequality constraint functions  $f_1, \dots, f_m$  are convex

the equality constraint functions  $h_i(\mathbf{x}) = a_i^T \mathbf{x} - b_i \forall i = 1, \dots, p$  are convex

The consequence of these qualifications is that the feasible set is convex. Thus, in a convex optimization problem, we minimize a convex function over a convex set.

### C.3 Karush-Kuhn Tucker (KKT) Optimality Conditions

This section is based on [48] Sec. 5.5.3.

Let C.3 be the primal problem and C.4 be the corresponding dual problem.

$$\min_{\mathbf{x}} f_0(\mathbf{x}) \quad (\text{C.3a})$$

$$\text{s.t. } f_i(\mathbf{x}) \leq 0, i = 1, \dots, m \quad (\text{C.3b})$$

$$h_i(\mathbf{x}) = 0, i = 1, \dots, p \quad (\text{C.3c})$$

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \quad g(\boldsymbol{\lambda}, \boldsymbol{\mu}) := \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad (\text{C.4a})$$

$$\text{s.t.} \quad \boldsymbol{\lambda} \succeq \mathbf{0} \quad (\text{C.4b})$$

where:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f_0(\mathbf{x}) + \boldsymbol{\lambda}^T \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} + \boldsymbol{\mu}^T \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_p(\mathbf{x}) \end{bmatrix} \text{ is the Lagrangian function}$$

Assume that the constraint functions  $f_1, \dots, f_m, h_1, \dots, h_p$  are differentiable, but not necessarily convex.

Let  $\mathbf{x}^*$ ,  $\boldsymbol{\lambda}^*$  and  $\boldsymbol{\mu}^*$  be a (not necessarily unique) triplet of primal and dual optimal points with zero duality gap i.e. strong duality holds such that  $f(\mathbf{x}^*) = g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ . Since  $\mathbf{x}^*$  minimizes  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$  over  $\mathbf{x}$ , it follows that the gradient of the Lagrangian with respect to  $\mathbf{x}$  must vanish at  $\mathbf{x}^*$ :

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \nabla_{\mathbf{x}} f_0(\mathbf{x}^*) + \boldsymbol{\lambda}^{*T} \begin{bmatrix} \nabla_{\mathbf{x}} f_1(\mathbf{x}^*) \\ \vdots \\ \nabla_{\mathbf{x}} f_m(\mathbf{x}^*) \end{bmatrix} + \boldsymbol{\mu}^{*T} \begin{bmatrix} \nabla_{\mathbf{x}} h_1(\mathbf{x}^*) \\ \vdots \\ \nabla_{\mathbf{x}} h_p(\mathbf{x}^*) \end{bmatrix} = \mathbf{0} \quad (\text{C.5})$$

For *any* optimization problem with differentiable cost and constraint functions for which strong duality holds, any pair of primal and dual optimal points must satisfy the *Karush-Kuhn Tucker* (KKT) optimality conditions, given by [C.6](#):

$$f_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m \quad (\text{C.6a})$$

$$h_i(\mathbf{x}^*) = 0, i = 1, \dots, p \quad (\text{C.6b})$$

$$\lambda_i \geq 0, i = 1, \dots, m \quad (\text{C.6c})$$

$$\lambda_i f_i(\mathbf{x}^*) = 0, i = 1, \dots, m \quad (\text{C.6d})$$

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)(\mathbf{x}^*) = \mathbf{0} \quad (\text{C.6e})$$

When the primal problem [C.3](#) is convex i.e.  $f_i$  is convex for all  $i = 1, \dots, m$  and  $h_i$  is affine for all  $i = 1, \dots, p$ , any pair of primal-dual points that satisfy the KKT conditions are also primal and dual optimal with zero duality gap.

## C.4 Order of Convergence & Rate of Convergence

A convergent sequence  $x_n$  with a limit  $x^*$  is said to have an *order of convergence*  $q \geq 1$  and *rate of convergence*  $\mu$  if [C.7](#) holds.

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^q} = \mu \quad (\text{C.7})$$

where:

$q = 1$  describes *Q-linear convergence*

$q = 2$  describes *Q-quadratic convergence*

$q = 3$  describes *Q-cubic convergence*

## C.5 Regret

In game theory, the *regret* of a decision maker (an online optimization algorithm in our case) is defined as the difference between the total cost it has incurred and that of the best fixed decision in hindsight. A good online optimization algorithm minimizes the upper bound of the worst-case regret (see [53] Sec 1.1).

Let  $\mathcal{A}$  be an online algorithm,  $L_\tau$  be the cost function at the  $\tau^{\text{th}}$  iteration,  $\mathcal{L}$  be the bounded family of cost functions and  $\text{MaxIter}$  be a given maximum number of iterations.

The *instantaneous regret* at iteration  $\tau$  describes the loss incurred by evaluating the objective function at the real-time optimizer  $\mathbf{z}_\tau$  instead of the batch optimizers, and is given by [C.8]

$$r_\tau := L_\tau(\mathbf{z}_\tau) - \min_{\mathbf{z} \in \mathcal{Z}} L_\tau(\mathbf{z}) \geq 0 \quad (\text{C.8})$$

The *cumulative regret* of  $\mathcal{A}$  is the sum of the instantaneous regret values over  $\text{MaxIter}$  iterations, and is given by [C.9]

$$R_T(\mathcal{A}) := \sum_{\tau=1}^{\text{MaxIter}} r_\tau \quad (\text{C.9})$$

If we can prove that the cumulative regret is sublinear for algorithm  $\mathcal{A}$  as in Fig. [C.1] i.e. the rate of increase of cumulative regret decreases with increasing iteration number:

$$\lim_{\tau \rightarrow \infty} \frac{R_\tau}{\tau} = 0 \quad (\text{C.10})$$

this implies that averaged over time, algorithm  $\mathcal{A}$  performs just as well as the best fixed strategy in hindsight. We say that such an algorithm is *no regret*, because most of the time it will eventually converge to a close-to-optimal solution.

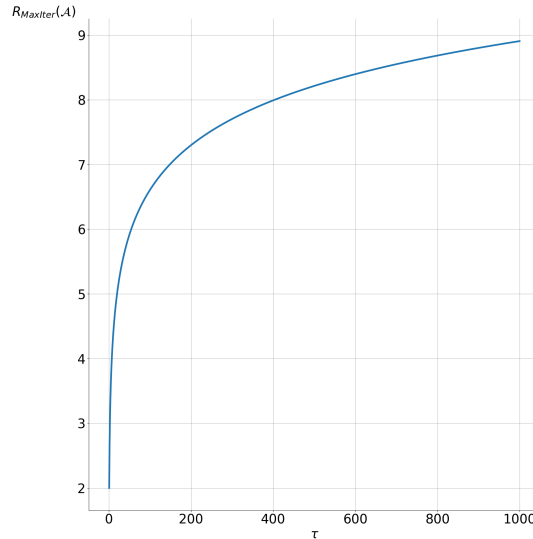


Figure C.1: Sublinear Cumulative Regret: if an algorithm has sublinear regret, it can be proven that the generated sequence of iterates eventually converges to the true optimizer of the objective function.

We also consider the *average regret* to compare regret to optimization error. Let  $\bar{\mathbf{z}} = \frac{1}{\text{MaxIter}} \sum_{\tau=1}^{\text{MaxIter}} \mathbf{z}_\tau$  be the *average decision*. Assuming that the functions  $L_1, \dots, L_{\text{MaxIter}}$  are all equal to a single real-valued convex function  $L : \mathcal{Z} \mapsto \mathbb{R}$ , then by Jensen's Inequality (defined in Appendix [D.8]):

$$\begin{aligned}
l\left(\frac{1}{\text{MaxIter}} \sum_{\tau=1}^{\text{MaxIter}} \mathbf{z}_\tau\right) &\leq \frac{1}{\text{MaxIter}} \sum_{\tau=1}^{\text{MaxIter}} l(\mathbf{z}_\tau) \\
l(\bar{\mathbf{z}}) - l(\mathbf{z}_*) &\leq \frac{1}{\text{MaxIter}} \sum_{\tau=1}^{\text{MaxIter}} [l(\mathbf{z}_\tau) - l(\mathbf{z}_*)] = \frac{r_\tau}{\text{MaxIter}}
\end{aligned} \tag{C.11}$$

which implies that  $l(\bar{\mathbf{z}})$  converges to  $l(\mathbf{z}_*)$  at a rate at most the average regret.

# Appendix D

## System & Control Theory

### D.1 Controllability, Observability & Stabilizability of a System

**Definition D.1.1** (Controllability, [44] Sec 1.3.4). *A system is controllable if, for any pair of states  $\mathbf{x}_1, \mathbf{x}_2$  in the state space,  $\mathbf{x}_2$  can be reached (or controlled to) in finite time from  $\mathbf{x}_1$ .*

**Definition D.1.2** (Observability, [44] Sec 1.4.5). *A system  $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w})$  is observable if there exists a finite horizon length  $N$ , such that for every initial state  $\mathbf{x}_0$ ,  $N$  measurements  $[\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{N-1}]$  distinguish uniquely the initial state  $\mathbf{x}_0$ .*

**Definition D.1.3** (Stabilizability, [44] Ex. 1.19). *A system  $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w})$  is stabilizable if any uncontrollable nodes (or states) are stable.*

### D.2 Discretizing a Nonlinear Continuous Dynamical System

For MPC and other digital control purposes the continuous-time (CT) dynamic state equations must be converted to discrete-time equations. In this section we describe two methods of discretizing a nonlinear continuous-time dynamical system.

Consider a continuous time-invariant system [D.1](#)

$$\begin{aligned}\dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \\ \mathbf{y} &= h(\mathbf{x}, \mathbf{u})\end{aligned}\tag{D.1}$$

with states  $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ , measured outputs  $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ , control inputs  $\mathbf{u}(t) \in \mathbb{R}^{n_u}$  and disturbances  $\mathbf{w}(t) \in \mathbb{R}^{n_w}$ . We define the discrete time index  $k$  such that  $t = k\Delta t$  where  $\Delta t$  is the sampling time. The time instances of relevance when the measured outputs will be *sampled* and the control inputs *applied and held* are then  $k\Delta t \mid k = 0, 1, \dots, N - 1$ . In general, the control input at time  $k\Delta t$ ,  $\mathbf{u}_k = \mathbf{u}(k\Delta t)$  will be held constant for the time interval  $t \in [k\Delta t, (k + 1)\Delta t)$  such that the continuous control inputs  $\mathbf{u}(t)$  relate to the discrete control inputs  $\mathbf{u}_k$  as:

$$\mathbf{u}(t) = \mathbf{u}_k, t \in [k\Delta t, (k + 1)\Delta t)\tag{D.2}$$

The states are modeled and the outputs measured at the same discrete time intervals such that the continuous states  $\mathbf{x}(t)$  and measured outputs  $\mathbf{y}(t)$  relate the discrete states and measured outputs as:

$$\mathbf{y}_k = \mathbf{y}(k\Delta t)\tag{D.3}$$

$$\mathbf{x}_k = \mathbf{x}(k\Delta t)\tag{D.4}$$

*Discretization* refers to the derivation of a discrete dynamic state equation from the continuous model:

$$\mathbf{x}_{k+1} = f^d(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)\tag{D.5}$$



### D.2.1 First-Order Taylor Series Discretization

The *Taylor Series first-order discretization* method approximates a constant rate of change  $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w})$  over the discretization sampling interval  $\Delta t$ , such that the discretized dynamic state equation is given by Eqn. [D.6](#).

$$\mathbf{x}_{\mathbf{k}+1} = \mathbf{x}_{\mathbf{k}} + \Delta t f(\mathbf{x}_{\mathbf{k}}, \mathbf{u}_{\mathbf{k}}, \mathbf{w}_{\mathbf{k}}) \quad (\text{D.6})$$

### D.2.2 Runge-Kutte Discretization

The fourth-order *Runge-Kutte* methods [54](#) are a generalization of the Euler discretization method in that they allow for multiple evaluations of the derivative in the neighborhood of the input variables, see [D.7](#).

$$k_1 = f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \quad (\text{D.7a})$$

$$k_2 = f\left(\mathbf{x} + \Delta t \frac{k_1}{2}, \mathbf{u}, \mathbf{w}\right) \quad (\text{D.7b})$$

$$k_3 = f\left(\mathbf{x} + \Delta t \frac{k_2}{2}, \mathbf{u}, \mathbf{w}\right) \quad (\text{D.7c})$$

$$k_4 = f(\mathbf{x} + \Delta t k_3, \mathbf{u}, \mathbf{w}) \quad (\text{D.7d})$$

$$\mathbf{x}_{\mathbf{k}+1} = \mathbf{x}_{\mathbf{k}} + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) \quad (\text{D.7e})$$

## D.3 Jensen's Inequality

Let a *Euclidean space* be defined as in Def. [D.3.1](#).

**Definition D.3.1** (Euclidean Space). A Euclidean space  $\mathbb{E}$  is a finite dimensional space endowed with an inner product  $\langle \cdot, \cdot \rangle$  and the Euclidean norm  $\|\cdot\| = \langle \cdot, \cdot \rangle$ .

**Definition D.3.2** (Jensen's Inequality). Jensen's Inequality is satisfied for a convex function  $f : \mathbb{E} \mapsto [-\infty, \infty)$  if it holds that:

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}) \quad (\text{D.8})$$

## D.4 Game Theory

### D.4.1 Stackelberg Game

A *Stackelberg game* consists of multiple players (or decision makers). One player (the *leader*) makes the first decision. The remaining players (the *followers*) observe Player 1's decision and subsequently make their decisions in parallel.

### D.4.2 Social Welfare Problem

In a *social welfare* optimization problem, the cost function consists of the sum of individual weighted utilities for each player and is maximized over all feasible allocations that satisfy certain constraints.

### D.4.3 Multi-Armed Bandit Problem

In a *multi-armed bandit problem*, a fixed and limited set of resources must be allocated between competing choices such that their expected gain is maximized, where each choice's properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating resources to the choice [55](#).

## D.5 Tikhonov Regularization

In the context of optimal control, *Tikhonov regularization* involves augmenting the cost function with the scaled Euclidean norm of the optimization variable vector. Let  $\mathbf{x}$  be a vector of optimization variables and  $f_0(\mathbf{x})$  be the cost function. The unconstrained optimal control problem is given by [D.9](#).

$$\min_{\mathbf{x}} f_0(\mathbf{x}) \tag{D.9}$$

The Tikhonov regularized equivalent is given by [D.10](#) for  $\delta > 0$ .

$$\min_{\mathbf{x}} f_0 + \delta \|\mathbf{x}\|_2^2 \tag{D.10}$$

# Bibliography

- [1] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-Based Model Predictive Control: Toward Safe Learning in Control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.
- [2] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P. Yves Glorennec, H. Hjalmarsson, and A. Juditsky, “Nonlinear Black-Box Modeling in System Identification: a Unified Overview,” *Automatica*, vol. 31, pp. 1691–1724, 1995.
- [3] A. Bemporad and M. Morari, “Robust Model Predictive Control: A Survey,” in *Robustness in Identification and Control*, 1998.
- [4] A. Mesbah, “Stochastic Model Predictive Control: An Overview and Perspectives for Future Research,” *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.
- [5] S. Bahrami, V. W. S. Wong, and J. Huang, “An Online Learning Algorithm for Demand Response in Smart Grid,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4712–4725, 2018.
- [6] A. Lesage-Landry and J. A. Taylor, “Setpoint Tracking With Partially Observed Loads,” *IEEE Transactions on Power Systems*, vol. 33, no. 5, pp. 5615–5627, 2018.
- [7] A. Lesage-Landry and D. S. Callaway, “Dynamic and Distributed Online Convex Optimization for Demand Response of Commercial Buildings,” *IEEE Control Systems Letters*, vol. 4, p. 632 637, Jul 2020.
- [8] P. Palensky and D. Dietrich, “Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 2011.
- [9] J. Su, Y. Jiang, A. Bitlislioglu, C. N. Jones, and B. Houska, “Distributed Multi-Building Coordination for Demand Response,” 2021.
- [10] N. Gatsis and G. B. Giannakis, “Residential Load Control: Distributed Scheduling and Convergence With Lost AMI Messages,” *IEEE Transactions on Smart Grid*, vol. 3, no. 2, pp. 770–786, 2012.
- [11] B. Lokeshgupta, A. Sadhukhan, and S. Sivasubramani, “Multi-Objective Optimization for Demand Side Management in a Smart Grid Environment,” in *2017 7th International Conference on Power Systems (ICPS)*, pp. 200–205, 2017.
- [12] J. Koshal, A. Nedic, Uday, and V. Shanbhag, “Multiuser Optimization: Distributed Algorithms and Error Analysis,” *SIAM J. Control Optim.*, pp. 1046–1081, 2011.
- [13] A. M. Ospina, A. Simonetto, and E. Dall’Anese, “Personalized demand response via shape-constrained online learning,” in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, 2020.

- [14] X. Zhou, E. Dall’Anese, and L. Chen, “Online Stochastic Optimization of Networked Distributed Energy Resources,” *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2387–2401, 2020.
- [15] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting,” *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.
- [16] F. Berkenkamp, A. P. Schoellig, and A. Krause, “No-Regret Bayesian Optimization with Unknown Hyperparameters,” 2019.
- [17] S. Van Vaerenbergh, J. Via, and I. Santamaría, “Nonlinear System Identification using a New Sliding-Window Kernel RLS Algorithm,” *Journal of Communications*, vol. 2, 05 2007.
- [18] Y. Engel, S. Mannor, and R. Meir, “The Kernel Recursive Least-Squares Algorithm,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.
- [19] L. Hoegaerts, L. De Lathauwer, I. Goethals, J. Suykens, J. Vandewalle, and B. De Moor, “Efficiently Updating and Tracking the Dominant Kernel Principal Components,” *Neural Networks*, vol. 20, no. 2, pp. 220–229, 2007.
- [20] J. Hanson, M. Raginsky, and E. Sontag, “Learning Recurrent Neural Net Models of Nonlinear Systems,” 2020.
- [21] D. Liao-McPherson, T. Skibik, J. Leung, I. Kolmanovsky, and M. M. Nicotra, “An Analysis of Closed-Loop Stability for Linear Model Predictive Control Based on Time-Distributed Optimization,” 2020.
- [22] A. Zanelli, Q. T. Dinh, and M. Diehl, “A Lyapunov Function for the Combined System-Optimizer Dynamics in Nonlinear Model Predictive Control,” 2020.
- [23] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, “Learning-based Model Predictive Control for Safe Exploration,” 2018.
- [24] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, “Learning and Control Using Gaussian Processes,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 140–149, 2018.
- [25] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, “Gaussian Process Model Based Predictive Control,” in *Proceedings of the 2004 American Control Conference*, vol. 3, pp. 2214–2219 vol.3, 2004.
- [26] F. Berkenkamp and A. P. Schoellig, “Safe and Robust Learning Control with Gaussian Processes,” in *2015 European Control Conference (ECC)*, pp. 2496–2501, 2015.
- [27] Y. Tang, E. Dall’Anese, A. Bernstein, and S. Low, “Running Primal-Dual Gradient Method for Time-Varying Nonconvex Problems,” 2018.
- [28] Y. Tang, *Time-Varying Optimization and Its Application to Power System Operation*. PhD thesis, 2019.
- [29] A. Bernstein, E. Dall’Anese, and A. Simonetto, “Online Primal-Dual Methods With Measurement Feedback for Time-Varying Convex Optimization,” *IEEE Transactions on Signal Processing*, vol. 67, no. 8, pp. 1978–1991, 2019.
- [30] T. Chen and G. B. Giannakis, “Bandit Convex Optimization for Scalable and Dynamic IoT Management,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1276–1286, 2019.
- [31] Y. Li, X. Chen, and N. Li, “Online Optimal Control with Linear Dynamics and Predictions: Algorithms and Regret Analysis,” 2019.

- [32] S. Koch, J. L. Mathieu, and D. S. Callaway, "Modeling and Control of Aggregated Heterogeneous Thermostatically Controlled Loads for Ancillary Services," in *in Proc. Power Systems Computation Conf*, 2011.
- [33] J. L. Mathieu, M. Kamgarpour, J. Lygeros, and D. S. Callaway, "Energy Arbitrage with Thermostatically Controlled Loads," in *2013 European Control Conference (ECC)*, pp. 2519–2526, IEEE, 2013.
- [34] J. L. Mathieu and D. S. Callaway, "State Estimation and Control of Heterogeneous Thermostatically Controlled Loads for Load Following," in *2012 45th Hawaii International Conference on System Sciences*, pp. 2002–2011, IEEE, 2012.
- [35] M. Kamgarpour, C. Ellen, S. E. Z. Soudjani, S. Gerwin, J. L. Mathieu, N. Müllner, A. Abate, D. S. Callaway, M. Fränzle, and J. Lygeros, "Modeling Options for Demand Side Participation of Thermostatically Controlled Loads," in *2013 IREP Symposium Bulk Power System Dynamics and Control-IX Optimization, Security and Control of the Emerging Power Grid*, pp. 1–15, IEEE, 2013.
- [36] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," 2010.
- [37] S. J. Gershman and D. M. Blei, "A Tutorial on Bayesian Nonparametric Models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, 2012.
- [38] C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [39] P. Bartlett, "Reproducing Kernel Hilbert Spaces," 2008.
- [40] R. Neal, "Lecture Notes in Statistics," *Bayesian Learning for Neural Networks*, 1996.
- [41] A. McHutchon, "Differentiating Gaussian Processes," 2013.
- [42] N. Aronszajn, "Theory of Reproducing Kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.
- [43] T. X. Nghiem and C. N. Jones, "Data-driven demand response modeling and control of buildings with Gaussian Processes," in *2017 American Control Conference (ACC)*, pp. 2919–2924, 2017.
- [44] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.
- [45] E. Dall'Anese, A. Simonetto, S. Becker, and L. Madden, "Optimization and Learning With Information Streams: Time-varying algorithms and applications," *IEEE Signal Processing Magazine*, vol. 37, p. 71–83, May 2020.
- [46] A. Beck, *First-Order Methods in Optimization*. SIAM, 2017.
- [47] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. SIAM, 2014.
- [48] S. Boyd and L. Vandenberghe, *Convex Optimization*. 03 2004.
- [49] A. Beck and M. Teboulle, "A Conditional Gradient Method with Linear Rate of Convergence for Solving Convex Linear Systems," *Mathematical Methods of Operations Research*, p. 235–247, 2004.
- [50] B. Wittenmark, "Adaptive Dual Control Methods: An Overview," in *In 5th IFAC symposium on Adaptive Systems in Control and Signal Processing*, pp. 67–72, 1995.

- [51] A. Henry, “<https://github.com/achenry/online-gp-mpc>.”
- [52] “TCL Building Control Input-Output Data.”
- [53] E. Hazan, *Introduction to Online Convex Optimization*. 2019.
- [54] J. Butcher, “Numerical Methods for Ordinary Differential Equations,” pp. i–xxiv, 08 2016.
- [55] Z. Ying, “Multi-Armed Bandit Allocation Indices,” *Technometrics*, vol. 33, 03 2012.





Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

Automatic Control Laboratory

**Title of work:**

Real-Time Learning-Based Model Predictive Control:  
Online Algorithms and Applications in Energy Systems

**Thesis type and date:**

Master Thesis, March 2021

**Supervision:**

Prof. Emiliano Dall'Anese, Systems & Control, University of Colorado Boulder  
Prof. Florian Dörfler, Automatic Control Laboratory, ETH Zürich

**Student:**

Name: Aoife Henry  
E-mail: henrya@student.ethz.ch  
Legi-Nr.: 18-946-756  
Semester: 5

**Statement regarding plagiarism:**

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

[http://www.ethz.ch/faculty/exams/plagiarism/confirmation\\_en.pdf](http://www.ethz.ch/faculty/exams/plagiarism/confirmation_en.pdf)

Boulder CO, 20. 3. 2021:





## Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

---

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

**Authored by** (in block letters):

*For papers written by groups the names of all authors are required.*

**Name(s):**

**First name(s):**

.....	.....
.....	.....
.....	.....
.....	.....

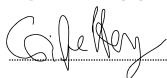
With my signature I confirm that

- I have committed none of the forms of plagiarism described in the '[Citation etiquette](#)' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

**Place, date**

**Signature(s)**

.....	
.....	.....
.....	.....
.....	.....

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*