ADAPTIVE MULTI-OBJECTIVE OPTIMIZING FLIGHT CONTROLLER

By

Ali Abdollahi

Bachelor of Science in Mechanical Engineering

Amirkabir University of Technology

Tehran, Iran

2013

Bachelor of Science in Biomedical Engineering

Amirkabir University of Technology

Tehran, Iran

2013

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2016

ADAPTIVE MULTI-OBJECTIVE OPTIMIZING FLIGHT CONTROLLER

Thesis Approved:

Dr. Girish Chowdhary

Committee Chair and Thesis Advisor

Dr. R. Russell Rhinehart

Committee Member

Dr. He Bai

Committee Member

Acknowledgments

This work was supported in part by University of Oklahoma NASA Cooperative Agreement No. NNX13AB21A and AFOSR Young Investigator Award No. FA9550-15-1-0146.

Name: ALI ABDOLLAHI

Date of Degree: JULY, 2016

Title of Study: ADAPTIVE MULTI-OBJECTIVE OPTIMIZING FLIGHT CONTROLLER

Major Field: MECHANICAL AND AEROSPACE ENGINEERING

Abstract: The problem of synthesizing online optimal flight controllers in the presence of multiple objectives is considered. A hybrid adaptive-optimal control architecture is presented, which is suitable for implementation on systems with fast, nonlinear and uncertain dynamics subject to constraints. The problem is cast as an adaptive Multi-Objective Optimization (MO-Op) flight control problem wherein control policy is sought that attempts to optimize over multiple, sometimes conflicting objectives. A solution strategy utilizing Gaussian Process (GP)-based adaptive-optimal control is presented, in which the system uncertainties are learned with an online updated budgeted GP. The mean of the GP is used to feedback-linearize the system and reference model shaping Model Predictive Control (MPC) is utilized for optimization. To make the MO-Op problem online realizable, a relaxation strategy that poses some objectives as adaptively updated soft constraints is proposed. The strategy is validated on a nonlinear roll dynamics model with simulated state-dependent flexible-rigid mode interaction. In order to demonstrate low probability of failure in the presence of stochastic uncertainty and state constraints, we can take advantage of chance-constrained programming in Model Predictive Control. The results for the single objective case of chance-constrained MPC is also shown to reflect the low probability of constraint violation in safety critical systems such as aircrafts. Optimizing the system over multiple objectives is only one application of the adapive-optimal controller. Another application we considered using the adaptive-optimal controller setup is to design an architecture capable of adapting to the dynamics of different aerospace platforms. This architecture brings together three key elements, MPC-based reference command shaping, Gaussian Process (GP)-based Bayesian nonparametric Model Reference Adaptive Control (MRAC) which both were used in the previous application as well, and online GP clustering over nonstationary (time-varying) GPs. The key salient feature of our architecture is that not only can it detect changes, but it uses online GP clustering to enable the controller to utilize past learning of similar models to significantly reduce learning transients. Stability of the architecture is argued theoretically and performance is validated empirically. 331 words

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Problem Statement

1.1 Introduction

Fuel efficiency and environmental compatibility can both be significantly improved by utilizing lightweight and high aspect ratio wing design [1]. However, using lightweight or high aspect ratio constructions leads to structures with flexible modes, which can lead to increased component wear or decreased ride comfort. Hence, the main challenge in order to leverage lightweight or high aspect ratio construction, is in designing aircraft control systems that optimize control input over multiple, sometimes conflicting objectives, such as maintaining a wing profile that minimizes drag and alleviating gust loading (which can be contradictory since gust load factors increase with increasing airspeed and a higher airspeed will result in higher drag), dampening out flexible modes to improve ride comfort, and ensuring that the aircraft has good handling qualities when tracking pilot commands. Also, gusts can cause ride discomfort showing the objectives inconsistency.

Decision-making problems that optimize over multiple objectives simultaneously are generally known as Multi-Objective Optimization (MO-Op) problems; The literature is reviewed briefly in Related Work section and more in detail in section 2.1. A characteristic of these types of problems is that there does not always exist a unique solution, rather a number of mathematically equally good solutions called Pareto optimal solutions [2, 3]. The set of feasible solutions is not explicitly known in advance for these problems, but it can be contained by constraint functions. There are two key challenges in developing Multi-Objective Optimization (MO-Op) flight control. Firstly, the dynamic model of the aircraft may not always be known in advance, especially in presence of intricate coupling between flexible and rigid body modes. Furthermore, even if such a model is known or can be learned using online data, a feasible optimal solution might be difficult to find online in the presence of onboard computational limitations.

The main goal for this problem is to describe a framework that can be utilized for adaptive MO-Op flight control by integrating learning-focused Model Reference Adaptive Control (MRAC) with constrained reference command shaping Model Predictive Control (MPC) introduced in [4, 5]. This integration allows us to optimize multiple objectives for flight control with relatively less computational effort by solving the optimal control problem over only the adaptive controller's reference model; while the adaptive part of the architecture learns the dynamics of the system that are unknown a-priori. The assumption is, that if the adaptive controller is able to make the plant behave like the reference model, and if the reference model is tracked exponentially, then the plant recovers the properties of the reference model. In particular, we show that when optimizing to achieve good command tracking while minimizing oscillations due to flexible modes, the MO-Op problem can be relaxed by posing the latter MO-Op objective as adaptively updated soft state constraint. This makes the problem practically realizable online. The critical elements of the presented architecture are a learning engine (Gaussian Process-based regression is used here), an online realizable MO-Op strategy, and a feedback structure that ensures the system states are stable during adaptation.

A secondary goal for this problem is to briefly introduce terminology and tools from the MO-Op literature to the aerospace community so that the advantages of our selected optimization method in our simulation will be contrasted and justified. Our hope is that a concise review will inspire other lines of inquiry in MO-Op flight control design. Section 3.1.1 is dedicated towards this objective.

The problem of controlling dynamical systems in the presence of constraints has received much attention. The extensive literature on Model Predictive Control (MPC; see e.g. [6, 7, 8]) has focused on model-based techniques for optimal control of constrained systems. For linear deterministic, and even for a large class of nonlinear deterministic systems, several MPC approaches exist, and have even been implemented on high-bandwidth platforms, such as quadrotors [9]. However, in the presence of stochastic modeling or actuation uncertainty, it is difficult to guarantee the abidance of state constraints with (linear) deterministic MPC approaches [10]. The typical approach of introducing a slack variable in these schemes cannot account for the state constraint violation. While the general problem of strict abidance of state constraints in the presence of stochastic uncertainty is hard, it is also true that most safety critical systems are often associated with an acceptable failure probability. For example, for certification of flight control systems, the Federal Aviation Administration currently requires that the probability of all catastrophic failures be limited to a small number (typically 10^{-9}). Chance-Constrained Model Predictive Control (CCMPC; see e.g. [10, 11]) is one approach for relaxing stochastic state-constrained MPC problems by allowing the constraints to be met with some level of probability. However, most current work in CCMPC assumes precise knowledge about the underlying model and is therefore not suited for uncertain systems with significant modeling uncertainty. That is why it should be incorporated with the learning-focused MRAC as part of the reference model shaping MPC.

Model Reference Adaptive Control (MRAC; see e.g. [12, 13, 14]) is one control methodology that tries to make an uncertain system behave like a-priori chosen reference dynamics. MRAC methods attempt to cancel the underlying nonlinear system uncertainties in order to ensure robust tracking of the designer-chosen reference model. To capture desirable robustness metrics, such as phase and gain margin, and transient response often a linear reference model is sought. This results in the reference trajectories being overly conservative, as they are not optimized to exploit the complete capabilities of the system. Shaping the reference command in order to account for input and state constraints has become known as the reference governor [15, 16, 17]. However, these methods often assume the plant to be linear, deterministic or require the existence of a stabilizing controller a-priori. Controlled physical systems are non-deterministic; in the presence of stochastic uncertainty, state constraints are frequently violated, thus rendering the MPC approach unreliable. So, the MPC part of our framework can be extended to CCMPC to accommodate stochastic uncertainties.

Control of aerospace vehicles that are subject to dynamic change is a challenging problem which motivates us to take on another extension of the adaptive-optimal control setup to address this issue. Such change could happen due to faults, reconfiguration of the vehicle, or due to different operating environments. A widely-employed method in aerospace controls is to first formulate a first-principles-based model of the system, identify its parameters through system identification, and then optimize the control design utilizing these models. However, in presence of nonstationarity induced due to dynamic change, a single-parameterized model may not be sufficient to describe the dynamics of the system. In fact, when the change is unexpected or sudden, such as due to a fault, a-priori models may not be applicable at all. Furthermore, the rapid growth in the number of existing and planned Unmanned Aerial System (UAS) platforms, many with highly modular payload capacity, is creating new control challenges. Consider for example the challenge scenario of designing an autopilot system that works in a plug-and-play manner – without requiring excessive tuning or system identification – across several Unmanned Aerial Vehicles (Figure 1.1) which has some parts in common with our challenge in this work. Ideally, such a system would be able to adapt to the dynamics of different aerospace platforms utilizing online-obtained data and generate optimal control policies over state and input constraints for these systems. However, it would also be desirable for the system to utilize past experience over similar platforms to quickly recognize similarities and use that knowledge to minimize the time spent in learning.

One way to deal with change is to estimate the parameters of a model online, and then utilize online optimization techniques, such as MPC to learn new control policies. The attractiveness of utilizing online MPC is in MPC's ability to optimize over state and input constraints. These methods, often referred to as indirect adaptive control mostly assume that the online-generated estimates are equal to the real parameters (an assumption known as certainty equivalence [18]). Several authors have studied adaptive-MPC architectures

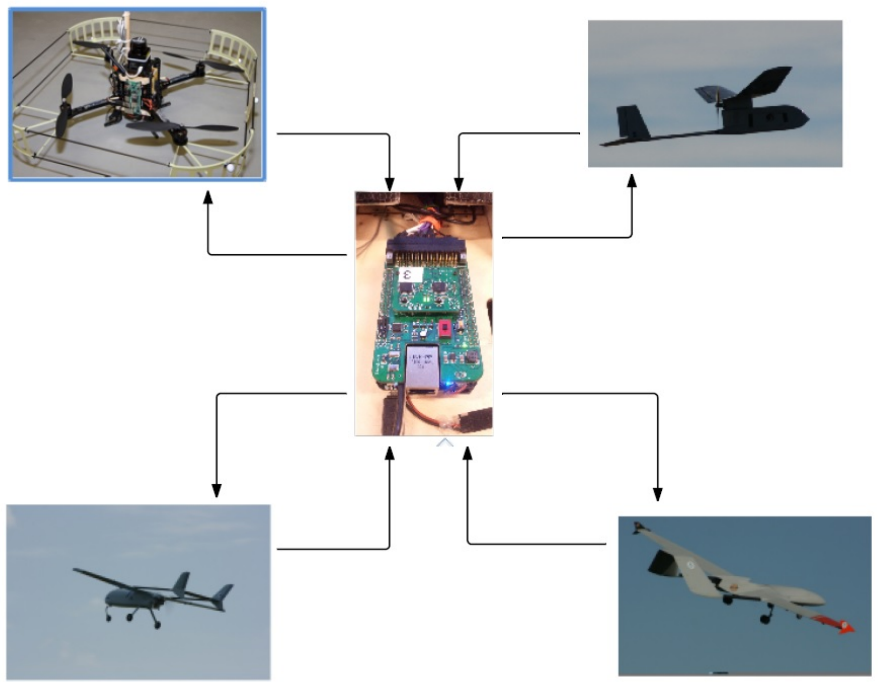


Figure 1.1: The challenge scenario of designing a plug-and-play autopilot.

that rely on variants of the certainty equivalence principle [9, 19, 20]. However, it is difficult to guarantee stability of such methods, especially during parameter estimation transients [21].

On the other hand, direct adaptive controllers such as MRAC can guarantee stability, even during harsh transients. These techniques have been successfully employed for flight vehicle control [22]. However, MRAC methods do not typically provide stability under state constraints. Adaptive control literature also consists of hybrid direct-indirect control architectures [23, 24, 25]. These methods seek to employ learning in a direct adaptive control framework. For example, in Chowdhary et al.'s concurrent learning method, instantaneous data is used concurrently with specifically online-recorded data to speed up the learning of unknown system parameters [26]. The learning capability of concurrent learning has been utilized with MPC architectures to yield adaptive-optimal controllers [4, 27]. However, the main limitation of the concurrent learning approach, and in fact many other adaptive control and online system identification approaches, is that they utilize an a-priori fixed structure for modeling the unknown system dynamics, the parameters of which are learned online. On the other hand, the Gaussian Process MRAC (GP-MRAC) approach does not need to assume an a-priori model structure, rather, it utilizes online data to simultaneously infer both the structure and the parameters of the unknown system dynamics [28, 29].

The main challenge here is of adaptive control in presence of unforeseen changes in the system dynamics. Our architecture is inspired by Muhlegg et al.'s concurrent learning adaptive-optimal control architecture

[4] since it also relies on optimizing the reference model commands. This work extends MPC-based reference command shaping technique [4] to the Bayesian Nonparametric (BNP) adaptive control of Gaussian Processes [30]. Another contribution is in showing that by optimizing the reference model commands using MPC, the system performance can be optimized without losing the desirable stability guarantees of GP-MRAC. An online clustering method is utilized to identify and leverage similarities between online-learned GP models of the unknown dynamics. Therefore, the key salient feature of our architecture is that not only can it detect changes, but it also uses online GP clustering [31] to enable the controller to utilize past learning of similar models to significantly reduce learning transients.

1.1.1 Related Work

Model Predictive Control (MPC), also referred to as Receding Horizon Control and Moving Horizon Optimal Control, has been widely adopted in industry as an effective means to deal with multivariable constrained control problems [32, 33]. With advent in compact computing resources, authors have recently proposed and employed MPC architectures for real-time control of aerospace vehicles [34]. However, the performance of MPC can depend on how accurate the employed predictive model of the system dynamics is. Several authors have proposed learning-based MPC algorithms [9, 19, 20], however the presence of learning transients typically prevents a non-conservative solution to be formed. The Concurrent Learning Model Predictive Control (CL-MPC) architecture addressed this issue by bringing together learning-based MRAC and MPC and utilizes an online threshold test to switch between the two [27, 35]. However, CL-MPC is developed in the deterministic setting and it requires online optimization of the feedback-linearized system model. The latter represents one of the main challenges in implementing MPC-based architectures: guaranteeing computational feasibility of obtaining an optimal solution online. Muhlegg et al. in [4] tackled this problem by solving offline the optimal control problem for a linear reference model that the adaptive controller is guaranteed to track. Inspired by [4], our main contributions include the use of Bayesian Non-Parametric (BNP) Gaussian Process models of the uncertainty that require very few assumptions about the unknown system dynamics a-priori.

Multi-Objective Optimization has been studied for several decades based on the theoretical background established in 1890's [36]. In a restricted sense, the Linear Quadratic Regulator (LQR), in which the control objective is to both reduce the tracking error and minimize required control, can be considered as a MO-Op problem posed by linearly combining multiple control objectives. More general formulations and solution approaches to MO-Op have been founded on the theory of mathematical programming which are explained in section 2.1 with their advantages and disadvantages for flight control problems. The key challenge with

MO-Op is that there may exist multiple mathematically equally good solutions, known as Pareto optimal solutions. Many MO-Op methods rely on the human in the loop to discriminate between these solutions [3]. There are some works done with the Multi-Objective Optimization method in a control setting which are fairly different with what we seek here. Nguyen developed a multi-objective optimal control modification for adaptive control of systems with both input and unmatched uncertainty [37]. The work done by Joos in [38] is multi-objective assessment in flight control law design. Nguyen et al. introduced a multi-objective linear quadratic optimal control approach that includes objective functions for drag and aeroelastic states in the cost function to design drag-cognizant flight control for flexible aircraft [1, 39]. Sadid et al. [40] used Multi-Objective Optimization for control of discrete-event systems to have a trade-off between cost and accuracy. In [41], Gambier designed his controller gains with Multi-Objective Optimization to avoid the control sensitivity to parameter changes of the plant. Somewhat similar to this idea, Fravolini et al. [42] computed their control strategy parameters with Multi-Objective Optimization while expressing the design requirements in terms of matrix inequalities. Using matrix inequalities, Swei et al. [43] designed their flexible aircraft controller with Multi-Objective Optimization. In general, there has been very little work in utilizing MO-Op strategies in an online setting, and we are not aware of dedicated work on online optimization-based MO-Op in flight control settings. Our main contribution in this area is to show that a class of MO-Op flight control problems can be relaxed by posing some of the control objectives as online updated soft constraints.

For the second extension of the framework, most existing GP inference algorithms assume that the underlying generative model is stationary. Grande et al.'s Gaussian Process Non-Bayesian Clustering (GP-NBC) algorithm [31, 44] decouples the problem of change-point detection, regression and model reuse, resulting in efficient online learning in the presence of change-points. GP-NBC is highly computationally efficient and orders of magnitude faster than other GP clustering methods, such as Dirichlet Process-based GP clustering [45]. We utilized GP-NBC in the above-mentioned adaptive-optimal control framework. We show that the resulting control architecture is guaranteed to be stable, and the likelihood of wrong clustering as well as the worst case tracking error due to misclassification are bounded.

1.2 Problem Formulation

We begin by describing the class of switching nonlinear dynamical systems our approach is applicable to. Let $x = [x_1 \;\; x_2]^T \in D_x \subset \mathbb{R}^l$ be defined as the state vector of the nonlinear system where D_x is a subset of $\mathbb{R}^l$. Let $u(t) \in \mathbb{R}^m$ be an admissible control input to the system, then the class of dynamical systems considered herein has the form:

$$\dot{x}(t) = Ax(t) + Bu(t) + B\Delta_i(x(t)). \tag{1.1}$$

where the state and input matrices $A \in \mathbb{R}^{l\times l}$ and $B \in \mathbb{R}^{l\times m}$ are the linear known components, and $\Delta_i(x(t)) \in \mathbb{R}^m$ is the unknown and switching component in the system with the switching index i. The solution to (1.1) induces a nonstationary continuous-time Markovian process. The process is nonstationary, i.e. it does not have a time-invariant generating distribution, because the unknown component Δ_i switches. We assume that the system (A, B) is fully controllable and the control input $u(t) \in \mathbb{R}^m$ is constrained due to limited control energy available to the system. It is further assumed that Δ_i is a stochastic process, and in particular a Gaussian Process (GP). Here we leverage the idea of modeling stochastic uncertainties $\Delta_i(x)$ in MRAC using Gaussian Processes [46]. That is,

$$\Delta_i(x) \sim \mathcal{GP}(m_i(x), K_i(x, x')). \tag{1.2}$$

where $m(\cdot)$ is the mean function of the GP and $K(\cdot,\cdot)$ is a real-valued covariance function. This model of the uncertainty has the advantage of being able to generate the appropriate model from data without assuming any prior knowledge over the domain [46]. In this treatment, the uncertainty is estimated as a sampled draw from a GP trained on a set of observed measurements. Note that we assume the mean function of a GP is Lipschitz continuous and the draw from a GP is a continuous function. Hence a unique solution to (1.1), x, exists for each switching index i. Furthermore, the nonlinear uncertainty $\Delta(x)$ is assumed to be a fixed unknown component in this work. The readers are referred to [5] for extensions to the class of switching dynamical systems .

The problem we are interested in solving is tracking the states of a reference model, x_{rm}, in presence of the nonlinear uncertainty, $\Delta(x)$, which forms the main objective and it is possible to have other objectives here as well that form a Multi-Objective Optimization problem. The MO-Op problem is of the form

$$\begin{gathered} \min f(x) = [f_1(x), f_2(x), ..., f_k(x)]^T. \\ \text{subject to } x \in S \end{gathered} \tag{1.3}$$

where we have $k \geq 2$ potentially conflicting objective functions $f_i \colon \mathbb{R}^n \to \mathbb{R}$ that we wish to minimize simultaneously. The decision vector $x = (x_1, x_2, ..., x_n)^T$ belongs to the non-empty feasible region $S \subset \mathbb{R}^n$. The vector function $f \colon \mathbb{R}^n \to \mathbb{R}^k$ is composed by k scalar objective functions. In MO-Op, the sets $\mathbb{R}^n$ and $\mathbb{R}^k$ are respectively known as decision variable space and objective space; they can also be defined as subsets of $\mathbb{R}^n$ and $\mathbb{R}^k$. The decision vector is regarded as optimal in MO-Op if they satisfy (1.3). Having the general form of MO-Op problem shown above, the tracking acts as only one of the objectives, f_i, if there are multiple objectives to optimize. The details about a MO-Op problem are discussed in section 2.1.

Continuing with the reference model, let $x_{rm}(t) \in D_x \subset \mathbb{R}^l$ be the state of the reference model which characterizes the desired response of the system. The dynamics of the reference model are assumed to take on the form:

$$\dot{x}_{rm}(t) = A_{rm}x_{rm}(t) + B_{rm}u_{rm}(x(t), r(t)). \tag{1.4}$$

where r is the reference command, $A_{rm} \in \mathbb{R}^{l\times l}$ and $B_{rm} \in \mathbb{R}^{l\times m}$ and the reference model input $u_{rm} \in \mathbb{R}^m$. Synthesis of u_{rm} in case of shaping the reference command in an optimal sense is known as the reference governor problem [47]; Otherwise, u_{rm} can be the reference command itself. The objective is to design a controller which can achieve a closed-loop performance such that the plant model defined by (1.1) tracks the reference model given by (1.4). Therefore, a control law is proposed as (1.5) with a feedback term defined as $u_{fb} = K_m^T x$; $K_m \in \mathbb{R}^{l\times m}$, a linear feedforward term, $u_{ff} = K_b^T u_{rm}$; $K_b \in \mathbb{R}^{m\times m}$, and an adaptive element, u_{ad}, for the plant to have the desired behavior. This is expressed as:

$$u(t) = u_{ff}(t) + u_{fb}(t) - u_{ad}(t). \tag{1.5}$$

The gain matrices K_m and K_b are chosen such that $A_{rm} = A + BK_m^\top$ and $B_{rm} = BK_b$ [See (3.19)]. Define the tracking error as $e(t) = x(t) - x_{rm}(t)$. Then by using (1.1) and (1.4) and then substituting (1.5) into (1.1), we can obtain the tracking error dynamics as:

$$\dot{e}(t) = A_{rm}e(t) + B(\Delta(x(t)) - u_{ad}(t)). \tag{1.6}$$

Having the tracking error dynamics as the differential equation in (1.6) provides us with an exponential decaying behavior of error if the second term on the right goes to zero. The details about this are explained in section 4.1.

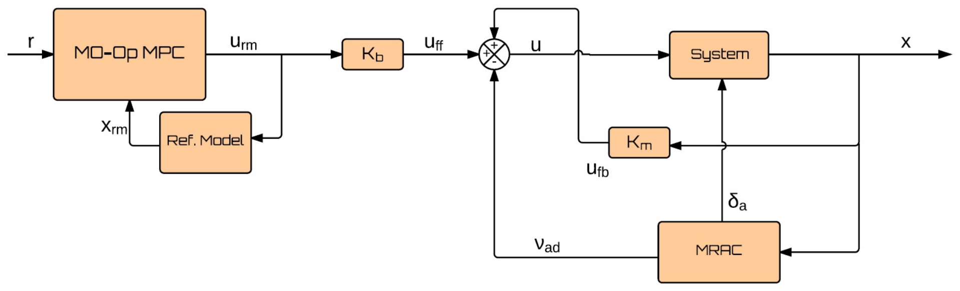


Figure 1.2: The proposed solution architecture for adaptive Multi-Objective Optimization.

We are now in a position to present an overview of the presented architecture shown in Figure 1.2. The architecture consists of two key elements, a reference command shaping MO-Op MPC and GP-MRAC-based adaptive control. The synthesis of u_{ad} is accomplished using the GP-MRAC method [30]. However, GP-MRAC is a tracking architecture that tracks a given reference command, it does not guarantee the optimality of the reference command, especially in presence of state and input constraints. An optimal reference command is a dynamically changing reference while applying some constraints. To accommodate this, we utilize the MPC-based reference command shaping technique proposed by Muhlegg et al. [4] and modify it such that we could incorporate multiple objectives therein. Hence, in our architecture, the reference command is shaped using MO-Op MPC on the reference model (1.4) in presence of state and input constraints optimizing over two objectives i.e. tracking error and oscillation. The optimized reference command is tracked by the GP-MRAC which simultaneously tracks the reference commands and learns a model of the underlying uncertainty $\Delta(x)$.

Now, we can present an overview of the second extension of the adaptive-optimal framework, shown in Figure 1.3, which is the solution to the problem of classifying different models.

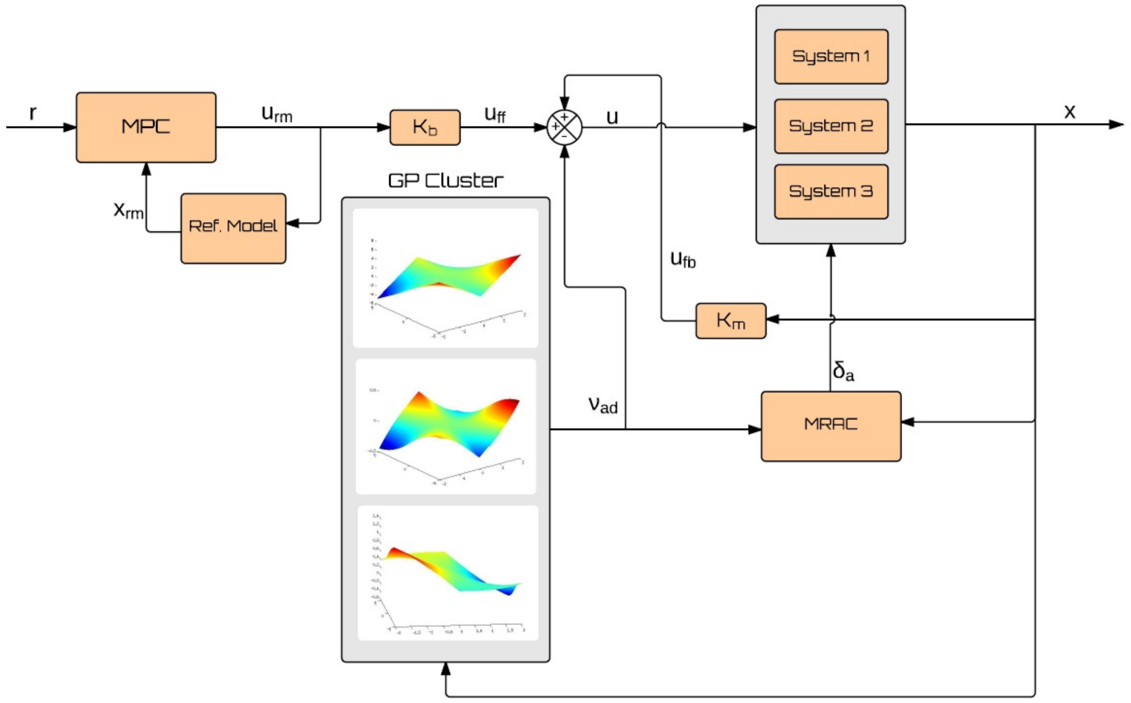


Figure 1.3: The proposed solution architecture for adaptive-optimal clustering.

The architecture consists of three key elements, a reference command shaping MPC, GP-MRAC-based adaptive control, and GP-NBC-based clustering. The synthesis of u_{ad} is accomplished using the GP-MRAC method again. The reference command is shaped using MPC on the linear reference model (1.4) in presence of state and input constraints. The optimized reference command is tracked by the GP-MRAC which simultaneously tracks the reference commands and learns a model of the underlying uncertainty Δ_i. The GP-NBC clustering algorithm utilizes online data to learn an estimate of a set of functions that contains the different means m_i and the associated hyperparameters of the kernels k_i by learning new GPs and clustering together those GPs from which samples have been previously seen. The stability of the entire architecture is argued in Section 4.1 and results are presented in Section 5.2.

CHAPTER 2

Multi-Objective Optimization

2.1 Multi-Objective Optimization (MO-Op)

As the general form of Multi-Objective Optimization problem is mentioned in the previous section, further mathematical details and a survey of solution methods in literature are provided here.

Pareto Optimality

In single-objective optimization, the relation "less than or equal ($\leq$)" is used to compare the values of the scalar objective functions. In contrast, in multi-objective problems, the objective function lies in $\mathbb{R}^k$. Since there is no canonical order on $\mathbb{R}^k$, other definitions of order are needed to compare vectors in $\mathbb{R}^k$.

In MO-Op, the Pareto dominance relation is usually adopted, originally proposed by Edgeworth in [48] but generalized by French-Italian economist, Vilfredo Pareto in [49].

Pareto Dominance Relation: We say that a vector z^1 Pareto dominates vector z^2, denoted by $z^1 \prec z^2$ (without loss of generality, only for the minimization case), if and only if:

$$\begin{gathered} \forall i \in \{1,...,k\} : z_i^1 \leq z_i^2 \\ \text{and} \\ \exists i \in \{1,...,k\} : z_i^1 < z_i^2 \end{gathered} \tag{2.1}$$

Pareto Optimality: In MO-Op, a decision vector $x^* \in S \subset \mathbb{R}^n$ is called Pareto Optimal if there does not exist another $x \in S$ such that $f(x) \prec f(x^*)$.

Weak Pareto Optimality: A solution $x^* \in S$, is weakly Pareto optimal if there does not exist another solution $x \in S$ such that $f(x) < f(x^*)$ for all $i = 1,...,k$..

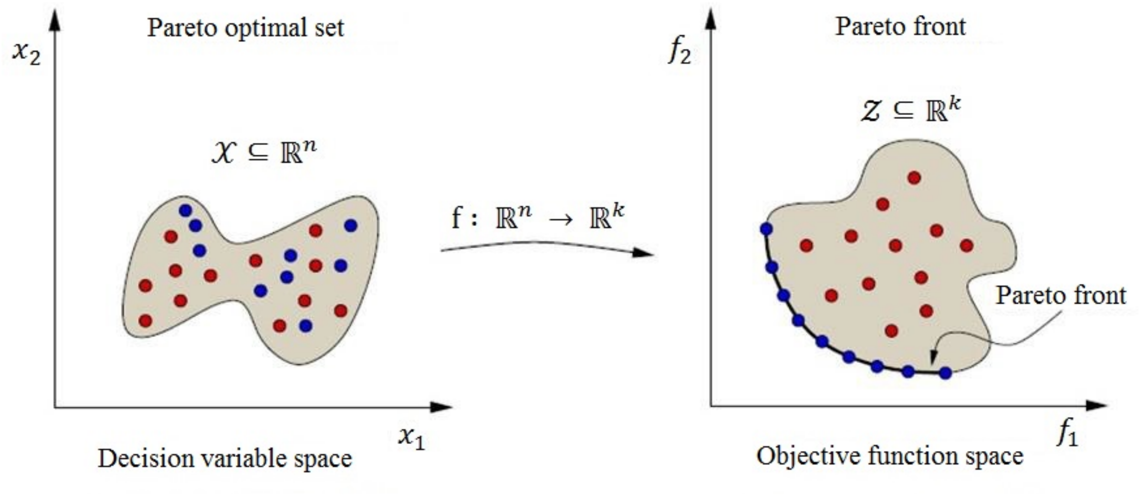

Pareto optimal set
x_2
$\mathcal{X} \subseteq \mathbb{R}^n$
x_1
Decision variable space
$\mathrm{f}: \mathbb{R}^n \rightarrow \mathbb{R}^k$
Pareto front
f_2
$\mathcal{Z} \subseteq \mathbb{R}^k$
Pareto front
f_1
Objective function space

Figure 2.1: The Pareto optimal set and its image, Pareto front.

Pareto Optimal Set: The Pareto optimal set, P^* is defined as:

$$P^* = \{x \in S \mid \nexists y \in S\colon f(y) \preceq f(x)\}. \tag{2.2}$$

Pareto Front: For a Pareto optimal set, P^*, the Pareto front, PF^*, is defined as:

$$PF^* = \{f(x) = [f_1(x), f_2(x), ..., f_k(x)]^\top \mid x \in P^*\}. \tag{2.3}$$

Figure 2.1 below illustrates the concept of Pareto optimal set and its image in the objective space, the Pareto front. Darker points denote Pareto optimal vectors. In variable space, these vectors are referred to as Pareto optimal decision vectors, while in objective space, they are called Pareto optimal objective vectors. As it is shown in Figure 2.1, the Pareto front is only composed by non-dominated vectors.

Mathematical Programming Techniques

The mathematical programming techniques are classified regarding how and when to incorporate preferences from the decision maker into the research process. Since mathematically the Pareto optimal decisions are equivalent, a decision maker is required to provide ordering. The decision maker is the person who shows his/her preferences among the objective and defines a tradeoff to be applied. A very important issue is the moment at which the decision maker is required to provide preference information. There are three ways of doing this [50]:

1) Prior to the search (a priori approaches)

2) During the search (interactive approaches)

3) After the search (a posteriori approaches).

Before we start with any of the categories, we introduce the global criterion method which is classified in some references as no-preference methods. This is suitable for situations when the decision maker does not have any special expectations of the solution and any optimal solution would be useful.

Global Criterion: This method is sometimes called compromise programming. In this method, the distance between some reference point and the feasible objective region is minimized. The reference point and a metric for distance measurement are chosen by user and there is no preference between objectives [51]. As an example, the ideal objective vector can be selected as the reference point and L_p-metrics are used as the distance metrics. So, the problem will take the form:

$$\begin{gathered} \min \Big(\sum_{i=1}^{k} |f_i(x) - z_i^*|^p \Big)^{1/p}. \\ \text{subject to } x \in S \end{gathered} \tag{2.4}$$

which z_i^* is the reference point. We know for the ideal objective vector that $f_i(x) \geq z_i^*$ for all $i = 1, ..., k$ and all $x \in S$.

If ideal objective vector is replaced by any other vector, it must not be pessimistic. Since this method cannot find solutions better than the reference point.

For $1 < p < \infty$, the exponent can be dropped, since L_p-metric is an increasing function. If $p = \infty$, the metric is called Tchebycheff metric which is to be discussed later.

Instead of the terms $|f_i(x) - z_i^*|$, denominators may be added to normalize the components since it may be useful to have the relative distances in calculations to reflect the location of ideal objective vector better. For example, using the components of z^* can shape the contour of the metric used.
It is obvious that this method is not useful for the MO-Op flight control problem because we do have preferences between our objectives and finding a proper reference point would also be an issue. With this method if we do not normalize the objective functions, an objective function whose ideal value is located nearer the feasible region would receive more importance.

Now, we discuss some of the well-known methods in the three categories mentioned before.

2.1. A Priori Methods

Value Function: In this method, the decision maker must be able to provide the explicit formulation of value function $U : \mathbb{R}^k \to \mathbb{R}$ that reflects the global preferences by giving a complete ordering in the objective space [51]. Then, the optimization problem

$$\begin{gathered} \max\ U(f(x)). \\ \text{subject to } x \in S \end{gathered} \tag{2.5}$$

can be solved by any single objective optimization method. Although, this seems like a simple method, the difficulty lies in the design of the mathematical expression of that value function that captures all the preferences of the decision maker exactly.

Goal Programming: Charnes and Cooper [52] are credited with the development of the goal programming method for a linear model, and played a key role in applying it to industrial problems. In this method, the decision maker has to assign targets or goals that they wish to achieve for each objective. These values are incorporated into the problem as additional constraints. The objective function then tries to minimize the absolute deviations from the targets to the objectives. The simplest form of this method may be formulated as follows:

$$\begin{gathered} \min \sum_{i=1}^{k} |f_i(x) - T_i| . \\ \text{subject to } x \in S \end{gathered} \tag{2.6}$$

where T_i denotes the target or goal set by the decision maker for the ith objective function. A more general formulation of the goal programming objective function is a weighted sum of the p^{th} power of the deviation $|f_i(x) - T_i|$. Such a formulation has been called generalized goal programming.
In the previous equation, the objective function is nonlinear and the simplex method can be applied only after transforming this equation into a linear form, thus reducing goal programming to a special type of linear programming. In this transformation, new variables δ_i^+ and δ_i^- are defined such that:

$$\begin{aligned} \delta_i^+ &= \tfrac{1}{2}\{|f_i(x) - T_i| + |f_i(x - T_i)|\}. \\ \delta_i^- &= \tfrac{1}{2}\{|f_i(x) - T_i| - |f_i(x - T_i)|\}. \end{aligned} \tag{2.7}$$

which are underachievement and overachievement variables. So, the resulting equivalent linear formulation is:

$$\begin{gathered} \min \sum_{i=1}^{k} (\delta_i^+ + \delta_i^-). \\ \text{subject to } f(x) - \delta_i^+ + \delta_i^- = T_i, i = 1, ..., k. \\ \delta_i^+, \delta_i^- \geq 0, i = 1, ..., k. \\ x \in S \end{gathered} \tag{2.8}$$

Since it is not possible to have both under and overachievements of the goal simultaneously, we have:

$$\delta_i^+ \cdot \delta_i^- = 0.$$

Although this is a very widely used solution method, the problem with this method is that in flight control problems, except for the tracking objective, we may not always know or wish to specify the target values for other objectives, rather it would be preferable to only minimize or maximize those objectives. Also, the underachievement and overachievement computation makes the problem computationally intensive, making

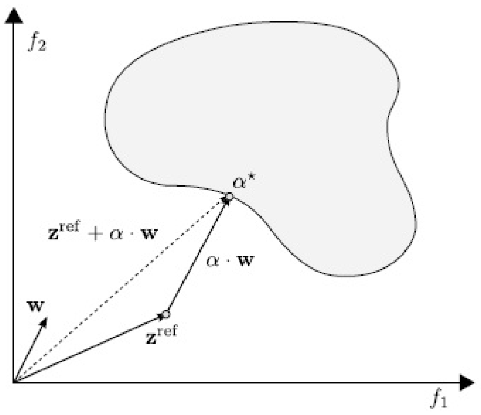

f_2
α^*
$\mathbf{z}^{\text{ref}} + \alpha \cdot \mathbf{w}$
$\alpha \cdot \mathbf{w}$
$\mathbf{w}$
$\mathbf{z}^{\text{ref}}$
f_1

Figure 2.2: Geometrical representation of goal attainment method for two objectives [54].

it difficult to implement online. The approach also requires significant decision maker involvement in the establishment of the targets and there is an underlying assumption of a piecewise linear value function.

Goal Attainment: In this method which is similar to goal programming, the decision maker must choose a goal vector, z^{ref} and a vector of weights, $W = [w_1, w_2, ..., w_k]$ to relate the under- and over-attainment of the desired goals [53]. To find the best-compromise solution, x^*, the problem is formulated as below:

$$\begin{aligned} &\min \alpha. \\ \text{subject to} \quad & z_i^{ref} + \alpha \cdot w_i \geq f_i(x); \quad i = 1, ..., k. \\ & x \in S \end{aligned} \tag{2.9}$$

where α is a scalar variable and the weights are normalized such that

$$\sum_{i=1}^{k} |w_i| = 1.$$

The term $\alpha \cdot w_i$ introduces an element of *slackness* into the problem, as opposed to imposing the goals rigidly. Hard constraints can be incorporated into the design by setting any of the weights to zero.

As shown in Figure 2.2, given z^{ref} and W, the direction for $z^{ref} + \alpha \cdot W$ vector is determined and the optimization problem stated in (2.9) finds the point on this vector in the feasible region which is closest to the origin. If this vector intersects the feasible region, it would clearly be a Pareto optimal solution. This optimum value of α reflects whether the goals are attainable or not. A negative value of α implies that the decision maker's goal is attainable. Otherwise, with $\alpha > 0$, it is not possible for the goal to be attained.

The same as goal programming method, the limiting factor here is choosing the proper z^{ref} and W. However, the built-in constraint relief is a useful feature of this approach.

Lexicographic Method: In this method, the objectives are arranged in order of importance by the decision maker (from most to least). The optimal value $f_i^*(i = 1, ..., k)$ is then obtained by minimizing the objective functions sequentially, starting with the most important one and proceeding according to the order of importance of the objectives. Additionally, the optimal value found for each objective, $f_i^*(x)$ is added as a constraint for subsequent optimizations. This way, the optimal value of the most important objectives is preserved.

$$\begin{aligned} &\min f_i(x) \\ \text{subject to} \quad & f_j(x) \leq f_j^*(x); \quad j = 1,2,...,i-1, \\ & i = 1,2,...,k. \\ & x \in S \end{aligned} \tag{2.10}$$

Only in the case of several optimal solutions in the single optimization of the current objective, the rest of the objectives will be considered. Since, only one of these solutions is Pareto optimal. Therefore, in the worst case, we have to carry out k single objective optimizations.

The disadvantages of this method might seem more evident. Firstly, this is a priority-driven method and the priority is rigid. So, if an objective cannot be optimized, the others will not be considered irrespectively, disregarding the possibility that they might be satisfied without any bad influence on the others. Additionally, if there are some objectives with equal priority or the number of priorities is less than the number of objectives [55], this method is ignorant of that fact and disregards all the other objectives with equal priority while optimizing the current one.

2.2. Interactive Methods

Geoffrion-Dyer-Feinberg (GDF): Geoffrion et al. [56] developed this method which is based on maximizing a utility function using a gradient-based method. This utility function is assumed to be differentiable and concave and it is implicitly known. There are some gradient-based methods to solve this problem such as Frank-Wolfe method [57]. Frank-Wolfe method assumes the feasible set S to be compact and convex. So, the direction-finding problem will be of the form

$$\begin{aligned} &\max \ \nabla_x U(f(x^h)) \cdot y. \\ &\text{subject to } y \in S \end{aligned} \tag{2.11}$$

where $U \colon \mathbb{R}^k \to \mathbb{R}$ is the utility function, x^h is the current point and y is the new variable of the problem. Using the chain rule for the gradient term and dividing it by $\frac{\partial U}{\partial f_1}$, the following formulation of Frank-Wolfe problem is obtained

$$\begin{aligned} &\max \ \Big(\sum_{i=1}^{k} -m_i^h \nabla_x f_i(x^h) \cdot y \Big). \\ &\text{subject to } y \in S \end{aligned} \tag{2.12}$$

where $m_i^h = \frac{\partial U}{\partial f_i} / \frac{\partial U}{\partial f_l}$ for all $i = 1,...,k$ are the marginal rates of indifference tradeoff at x^h between objectives f_l and f_i for $i \neq l$. Marginal rate of indifference tradeoff is the amount of loss on objective f_i that the decision maker is willing to tolerate in order to get a unit of gain in objective f_1, while there is no change in other objectives. Any of the other objectives can be the reference objective instead of f_1.

The GDF method procedure is as follows:

I. Provide an initial point x^h.

II. Provide marginal rates of indifference tradeoff between f_1 (the reference objective) and the other ones at current point x^h.

III. Find the optimal solution y^h to the problem (2.12). Set the new search direction as $d^h = y^h - x^h$. If $d^h = 0$, x^h is the final solution.

IV. Determine the best step size, t^h to compute the new solution, x^h. Set $x^{h+1} = x^h + t^h d^h$.

V. If $x^{h+1} = x^h$, x^h is the final solution, otherwise, set $h = h+1$ and go back to the first step.

To know the details of how step 2 and 4 are done, see [56].

As it is outlined in the description above, this method requires too much computation such as determining the $k-1$ marginal rates of substitution at each iteration apart from the utility function design with several mentioned assumptions and the difficult questions needed to be answered by the decision maker each time a new search direction is presented. These make it obsolete for the online control application. Also, there is no guarantee that the final solution is Pareto optimal.

Reference Point: This method is proposed by Wierzbicki [58] and its basic idea is as follows: First, the decision maker chooses a reference point. The aspiration levels for each objective is represented by this reference point. Then, the solutions that satisfy the aspiration levels the best are to be computed using an achievement scalarizing function. An achievement scalarizing function is a type of utility function, $s_{z^{ref}} : \mathbb{R}^k \to \mathbb{R}$ where $z^{ref} \in \mathbb{R}^k$ is the reference point showing the aspiration levels chosen by the decision maker. If the decision maker is satisfied with the current solution, the interactive optimization process will end. Otherwise, another reference point should be selected by the decision maker. Thus, the multi-objective problem follows as a scalar problem:

$$\begin{gathered} \min s_{z^{ref}}(z). \\ \text{subject to } z \in S \end{gathered} \tag{2.13}$$

The achievement scalarizing functions are mostly based on Tchebycheff metric (L_∞). One of the proper achievement scalarizing functions would be the augmented Tchebycheff scalarizing function which is defined below:

$$s_\infty(z, z^{ref}) = \max_{i=1,\dots,k} \{\omega_i(z_i - z_i^{ref})\} + \rho \sum_{i=1}^{k} \omega_i(z_i - z_i^{ref}). \tag{2.14}$$

where $\rho > 0$ is an augmentation coefficient sufficiently small and $\mathbf{W} = [\omega_1, ..., \omega_k]$ is a vector of weights such that $\forall i \ \omega_i \geq 0$ and $\exists i \ \omega_i > 0$.

Moving the reference point in each iteration makes the reference point methods explore the objective space. However, during the interactive process the weights do not change, since they do not define preferences. The weights main usage is to normalize each objective function. It is important to mention that the decision maker can choose both feasible or infeasible reference points. If the chosen reference point is feasible, the solution improves the aspiration levels. Otherwise, the solution will be the closest feasible point to the aspiration levels.

Choosing reference point as mentioned previously is one limitation for this method. As explained, this method does not help the decision maker find improved solutions while requiring him/her to answer difficult questions about current solution each time. In this approach only a finite subset of efficient alternatives can be considered and the final solution optimality cannot be checked. If the decision maker is not purposeful enough, the convergence is not necessarily fast.

Light Beam Search: The Light Beam Search (LBS) method proposed by Jaszkiewicz and Slowinski [59], is an iterative method which combines the tools of Multi-Attribute Decision Analysis (MADA) with reference point idea. The achievement function for minimization is the same as (2.14) where the weighting coefficients are only used in the 'max' part. Here, the reference point is assumed to be an infeasible objective and none of the objectives is more important than all the others together. Reference points make the decision maker able to guide the search process for solution. This process can be improved by providing the decision maker additional information about the Pareto optimal set at each iteration. So, a finite sample of nondominated solutions (a solution is nondominated if none of the objective functions can be improved at the cost of degrading some of the other objectives) is generated. This sample consists of the other solutions in the neighborhood of the current solution (based on the reference point method) which is called the middle point. To avoid the decision maker's frustration, concepts from multi-attribute decision analysis come into play. The idea is to establish outranking relations, S, between alternatives to define the neighborhood around the middle point. It is said that a outranks $b(aSb)$, if a is at least as good as b. Thus, several thresholds are required from the decision maker to define these outranking relations. The decision maker is asked to provide indifference thresholds for each objective function which are the intervals that indifference prevails. In order for the gap between indifference and preference not to be sharp, preference thresholds are expressed to show the hesitation interval. Another threshold called veto threshold is introduced to prevent a good performance in some objectives from compensating for poor values in others. Now it is possible for the decision maker to scan the neighborhood along the objective function trajectories between any two characteristic neighbors or between a characteristic neighbor and the middle point.

Main weakness of this method is its high computational demand to find the exact characteristic neighbors. Specifying different thresholds by the decision maker is a further demand too, although it is a progress not to assume them global.

The next three methods (in addition to lexicographic method) relax the problem in different ways so that it can be posed as a single objective optimization problem.

Tchebycheff Method: Tchebycheff method proposed in [60], is an iterative method that requires user inputs. This method minimizes a function value while assuming that the global ideal objective vector (Utopian vector, the vector that optimizes all the objective function which often does not exist) is known. Weighted Tchebycheff metric is the measurement metric used for distances to a utopian objective vector. Thus, the multi-objective optimization problem is formed as a single objective optimization problem:

$$\begin{gathered} \min \max[\omega_i(f_i(x) - z_i^*)], i = 1, ..., k. \\ \text{subject to } x \in S \end{gathered} \tag{2.15}$$

where $W = \{\omega \in \mathbb{R}^k \mid 0 < \omega_i < 1, \sum_{i=1}^{k} \omega_i = 1\}$ and z^* is the Utopian objective vector.

All the Pareto optimal solutions of the multi-objective optimization problem can be found by solving (2.15), however, some of the solutions to (2.15) might be weakly Pareto optimal ones. To avoid this disadvantage, we can formulate the problem as a lexicographic weighted Tchebycheff approach, as below:

$$\begin{gathered} \min \max_{i=1,\dots,k}[\omega_i(f_i(x) - z_i^*)], \sum_{i=1}^{k}(f_i(x) - z_i^*). \\ \text{subject to } x \in S \end{gathered} \tag{2.16}$$

Tchebycheff method generates a subset of nondominated solutions at each iteration. Among these solutions, there are the solutions generated by lexicographic weighted Tchebycheff method which are selected by the decision maker preference.

Flexibility of the method reduces by the fact that the discarded parts of the weighting vector space cannot be restored if the decision maker changes his/her mind. A great deal of calculation needed at each iteration of this method makes it an impractical choice. Also a lot of the results for the evaluation of objective functions is discarded at each step which might be laborious for complex problems.

2.3. A Posteriori Methods

Weighting Method: In this method, as presented in [61], the general idea is to assign each objective function a weighing coefficient and minimize the weighted sum of the objectives. In this way, the multi-objective problem is modified into the following single objective problem.

$$\begin{gathered}\min \sum_{i=1}^{k} \omega_i f_i(x). \\ \text{subject to } x \in S\end{gathered} \tag{2.17}$$

where $\omega_i \geq 0$ for all $i = 1, ..., k$ and is strictly positive for at least one objective, such that $\sum_{i=1}^{k} \omega_i = 1$. Linear Quadratic Regulators (LQR) utilize this technique for optimizing over both the control input and the regulation error.

Though this method is a simple way of finding the optimal solutions if the weighting coefficients are positive, all the Pareto optimal points cannot be found if the problem is non-convex. There might exist multiple minimum solutions for a specific weight vector which represent different solutions in the Pareto optimal front and this wastes the search effort. Also, two different set of weights do not necessarily lead to two different Pareto optimal solutions and an even distribution of weights among the objectives does not always result in an even distribution of solutions on the Pareto front. So, either too much simplicity or complication is not desired for the practical problems.

ε-constraint Method: In this approach, introduced in [62], one of the objective functions is selected based on a priority to be optimized and the other objectives are set as the constraints to the single objective optimization problem. The problem is of the form

$$\begin{gathered}\min f_l(x). \\ \text{subject to } f_j(x) \leq \varepsilon_j \text{ for all } j = 1, ..., k,\ j \neq l,\ \ x \in S\end{gathered} \tag{2.18}$$

where ε_j are upper bounds for the objectives. The idea is to relax the MO-Op problem by posing it as a constrained optimization problem.

Figure 2.3 shows the application of the ε-constraint method in the bicriterion problem. As shown below, there are three different upper bound values for objective function f_1 while we try to find the optimum value for f_2. Thus, depending on the upper bound selected, the respective optimum value can be different in each case.

The solution to ε-constraint problem (2.18) is weakly Pareto optimal in its general form. However, a decision vector $x^* \in S$ is Pareto optimal if and only if it is a solution of ε-constraint problem (2.18) where $\varepsilon_j = f_j(x^*)$ for all $j = 1, ..., k, j \neq l$.

Method of Weighted Metrics: Also called compromise programming [63], this method is the global criterion method with weighted metrics to produce different Pareto optimal solutions. So, by adding the

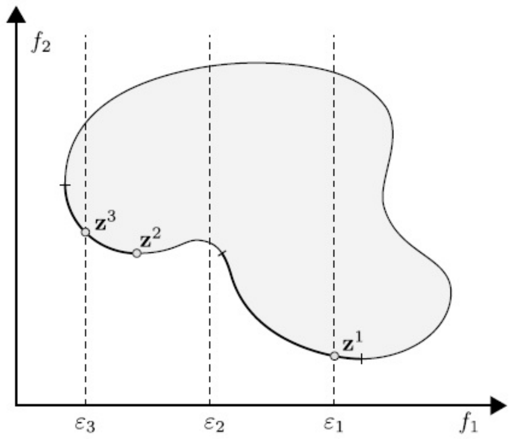


Figure 2.3: Illustration of the ε-constraint method [54].

weighting coefficients ω_i in the L_p-metric, we obtain the weighted L_p-problems:

$$\begin{gathered}\min \Big(\sum_{i=1}^{k} \omega_i \, |f_i(x) - z_i^*|^p \Big)^{1/p}. \\ \text{subject to } x \in S\end{gathered} \tag{2.19}$$

for $1 \le p < \infty$, where $\omega_i \ge 0$ for all $i = 1, ..., k$ and $\sum_{i=1}^{k} \omega_i = 1$.

The weighted Tchebycheff problem is then of the form:

$$\begin{gathered}\min \max_{i=1,...,k} \big(\omega_i \, |f_i(x) - z_i^*| \big). \\ \text{subject to } x \in S\end{gathered} \tag{2.20}$$

If the ideal objective vector is known globally, the absolute value signs can be dropped.

If $p = 1$, this method minimizes the sum of weighted deviations and it is equivalent to the weighting method except for the constant, z^* if it is known globally. For $p = 2$, this is the method of least squares. As p increases, the larger the weighted deviation becomes the more important its minimization is. Finally when p approaches to infinity, only the largest weighted deviation matters.

As mentioned in global criterion method, the requirement of the ideal solution, z^*, would be an issue. So, we need to optimize each objective function first to find these values. Another weakness is since different objectives may take on values of different orders of magnitude, it is advisable to normalize the objective functions but this requires the knowledge of minimum and maximum objective function values.

Normal Boundary Intersection: To distribute Pareto optimal solutions evenly, Das and Dennis in [64] suggested this method. Before describing the idea in this method, we define the pay-off matrix. Let x_i^* be the global minimizer of the respective objective function, $f_i(x), i = 1, ..., k$ in the feasible region, S and we assume $F(x_i^*) = F_i^*, i = 1, ..., k$, then the $k \times k$ matrix, Φ whose i^{th} column is $F_i^* - F^*$ is known as the pay-off matrix, where F^* is the utopian point defined by $F^* = \{f_1^*, ..., f_k^*\}$. Also, the set of points in $\mathbb{R}^k$ which are

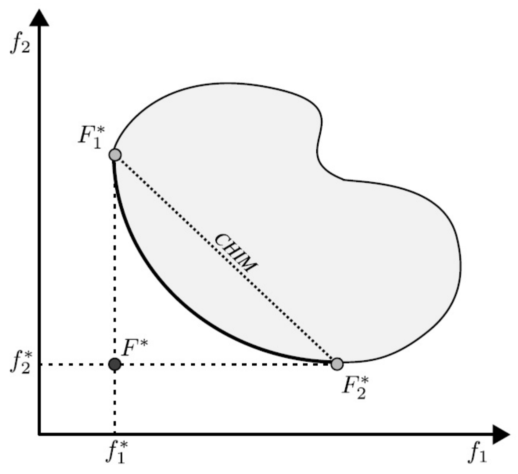


Figure 2.4: CHIM for a two-objective problem [54].

convex combination of these columns, i.e. $\{\Phi\beta : \beta \in \mathbb{R}^k, \sum_{i=1}^{k} \beta_i = 1, \beta_i \geq 0\}$, is named as Convex Hull of Individual Minima (CHIM). Now, the main idea in this method is to intersect a normal to CHIM with the feasible objective region, $\{F(x) : x \in S\}$.
Figure 2.4 shows the CHIM for a two-objective problem.

As it is shown in Figure 2.4, this approach finds a portion of the boundary of the feasible objective region which contains the Pareto optimal points by intersecting a normal pointing towards the origin from any point in the CHIM with this boundary. Thus, the original multi-objective optimization problem is transformed to the following:
Choose a weighting β as mentioned earlier in the definition of CHIM. Then, $\Phi\beta$ represents a point in the CHIM. If the unit normal to CHIM pointing towards the origin is defined by $\hat{n}$, then $\Phi\beta + t\hat{n}$ represents the set of points on the normal. So, the closest intersection of this normal and the boundary of the feasible objective region to the origin is generated by the global solution of the problem follows:

$$\begin{gathered} \max_{x,t}\ t. \\ \text{subject to } \Phi\beta + t\hat{n} = F(x) - F^* \\ x \in S \end{gathered} \tag{2.21}$$

Another way is to select a quasi-normal direction such that it forms an equally weighted linear combination of the columns of Φ.

$$\hat{n} = -\Phi v.$$

where v is a fixed vector with all the components positive and the minus sign is to ensure that it is pointing towards the origin.

This method is computationally costly and it involves many optimization subproblems. For further details regarding its limitations, see [65]. The computational cost is the main limiting factor preventing this method to be used for online control.

The different methods discussed above, each has their own benefits and disadvantages which are explained above briefly. Typically, the weighting method [61], and the ε-constraint [62] are more suitable for online autonomous decision-making. However, it is not always possible to find a linear combination of weights for obtaining non-conservative MO-Op solutions. Here we use the ε-constraint method since the constrained optimization problem is better suited for online flight control problems. This is because online flight control problems are often designed to accommodate other constraints most of the time, such as maximum allowable inputs or state constraints. Hence available constrained optimization routines such as quadratic optimization can be utilized. According to [66], this method is always able to identify the best compromise solutions belonging to the Pareto front independently of its shape and since the objective function bounds are known it can be used conveniently. Also, with this method we avoid finding the corner solutions (extreme solutions) to the problem and this makes the algorithm computationally efficient. In ε-constraint method, there is no need to scale the objectives to obtain the final solution and finally our MPC structure is simpler to implement and consistent with this solution method.

CHAPTER 3

Methods

3.1 Control Architecture

3.1.1 MO-Op Model Predictive Control (MPC)

The main idea in our architecture is to pose the MO-Op flight control problem as a single-objective constrained optimization problem (utilizing the ε-constraint method discussed in Section 2.1) which can then be solved online using well-studied techniques such as MPC. In this section a brief overview of MPC for constrained discrete-time systems is presented. The optimal reference command shaper is based on an explicitely generated offline solution to the MPC problem. The discretized version of the reference model dynamics is formulated as:

$$\begin{aligned} x_{rm}(k+1) &= A_{rm}x_{rm}(k) + B_{rm}u_{rm}(k), \\ y(k) &= C_{rm}x_{rm}(k). \end{aligned} \tag{3.1}$$

by taking a difference operation on both sides we have:

$$\begin{aligned} \Delta x_{rm}(k+1) &= A_{rm}\Delta x_{rm}(k) + B_{rm}\Delta u_{rm}(k), \\ \Delta y(k) &= C_{rm}\Delta x_{rm}(k). \end{aligned} \tag{3.2}$$

Defining $x(k) = [\Delta x_{rm}(k)^T \quad y(k)]^T$ results in the following state-space model:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u_{rm}(k), \\ y(k) &= Cx(k). \end{aligned} \tag{3.3}$$

where the triplet (A, B, C) which is called the augmented model, is as follows:

$$A = \begin{bmatrix} A_{rm} & o_{rm}^T \\ C_{rm}A_{rm} & 1 \end{bmatrix} B = \begin{bmatrix} B_{rm} \\ C_{rm}B_{rm} \end{bmatrix} C = \begin{bmatrix} o_{rm} & 1 \end{bmatrix}$$

where o_{rm} is a row vector of zeros with the same dimension as the state variable vector x_{rm}.

Upon formulation of the mathematical model, the next step in design of a predictive controller is to calculate the predicted plant output with the future control signal as the adjustable variables [67]. Assuming that at the sampling instant $k_i > 0$, the future control trajectory is denoted by:

$$\Delta u_{rm}(k_i), \Delta u_{rm}(k_i+1), \ldots, \Delta u_{rm}(k_i+N_c-1).$$

The future state variables are,

$$x(k_i+1 \mid k_i), x(k_i+2 \mid k_i), \ldots, x(k_i+N_p \mid k_i).$$

where N_p and N_c are prediction and control horizon, respectively. Based on the state-space model (3.3), the future state and output variables are calculated sequentially using the set of future control parameters [67] and form the compact matrix equation (3.4) by defining:

$$Y = \begin{bmatrix} y(k_i+1 \mid k_i) \\ y(k_i+2 \mid k_i) \\ \vdots \\ y(k_i+N_p \mid k_i) \end{bmatrix} \quad \Delta U = \begin{bmatrix} \Delta u_{rm}((k_i) \\ \Delta u_{rm}((k_i+1) \\ \vdots \\ \Delta u_{rm}((k_i+N_c-1) \end{bmatrix}$$

$$Y = Fx(k_i) + \phi \Delta U. \tag{3.4}$$

where,

$$F = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_p} \end{bmatrix} \quad \phi = \begin{bmatrix} CB & 0 & 0 & \ldots & 0 \\ CAB & CB & 0 & \ldots & 0 \\ CA^2B & CAB & CB & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \ldots & CA^{N_p-N_c}B \end{bmatrix}$$

For a given set-point signal $r(k_i)$ at sample time k_i, the optimization objective is to find the best control parameter vector ΔU such that an error function between the set-point and the predicted output is minimized.

Assuming that the data vector which contains the set-point information is $R_s^T = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix} r(k_i)$ with the dimension of N_p, the cost function J that reflects the control objective is defined as,

$$J = (R_s - Y)^T Q (R_s - Y) + \Delta U^T \bar{R} \Delta U. \tag{3.5}$$

Here, Q denotes a positive-definite diagonal matrix and $\bar{R} = r_w I_{N_c}$ $(r_w \geq 0)$ where r_w is a tuning parameter for desired closed-loop performance. The larger r_w is, the more important the size of ΔU will be. Conversely, the larger r_w becomes, the less important it becomes to make the error $(R_s - Y)^T(R_s - Y)$ small [67]. If we insert equation (3.4) into (3.5), we could form the MPC problem as minimizing the cost function,

$$\begin{aligned} J &= (R_s - Fx(k_i))^T Q(R_s - Fx(k_i)) \\ &- 2\Delta U^T \phi^T Q(R_s - Fx(k_i)) \\ &+ \Delta U^T(\phi^T Q\phi + \bar{R})\Delta U. \end{aligned} \tag{3.6}$$

which is subject to constraints on the control input and the output, that are collected together in equation (3.7) as a matrix compact form [67].

$$\begin{bmatrix} -C_2 \\ C_2 \\ -\phi \\ \phi \end{bmatrix} \Delta U \le \begin{bmatrix} -U_{min} + C_1 u_{rm}(k_i - 1) \\ U_{max} - C_1 u_{rm}(k_i - 1) \\ -Y_{min} + Fx(k_i) \\ Y_{max} - Fx(k_i) \end{bmatrix}. \tag{3.7}$$

where,

$$C_1 = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} \quad C_2 = \begin{bmatrix} I & 0 & \dots & \dots & 0 \\ I & I & 0 & \dots & 0 \\ & & \vdots & & \\ I & & \dots & & I \end{bmatrix}$$

In this work, as mentioned in the beginning of this section, the MO-Op flight control problem is to be cast as a single-objective constrained optimization problem (utilizing ε-constraint method) minimizing the cost function (3.5) to optimize the first objective, tracking error, while having constraints on the control input and the output as shown in (3.7). Our MO-Op MPC problem also optimizes the second objective, oscillation (energy in the system), by posing constraints on the system state. Note that the designer-chosen reference model is a second-order system in this work which the discrete-time state-space model is:

$$\begin{aligned} x_{rm_1}(k+1) &= x_{rm_2}(k), \\ x_{rm_2}(k+1) &= \omega_n^2\big(u_{rm}(k) - x_{rm_1}(k)\big) - 2\zeta\omega_n x_{rm_2}(k). \end{aligned} \tag{3.8}$$

To minimize the oscillation in a second-order system, it is possible to minimize the root mean square of the velocity $x_{rm_2}(k)$ at all time steps which results in bounding the velocity. Thus, a bound for the absolute value of velocity is assumed to derive the state constraints for optimizing the second objective. The velocity is calculated using the state-space model (3.8) as below:

$$x_{rm_2}(k) = \frac{-x_{rm_2}(k+1) + \omega_n^2\big(u_{rm}(k) - x_{rm_1}(k)\big)}{2\zeta\omega_n}.$$

Assume $|x_{rm_2}(k)| < \varepsilon$. Now solving for $u_{rm}(k)$ we have,

$$\frac{-2\zeta\omega_n\varepsilon + x_{rm_2}(k+1)}{\omega_n^2} + x_{rm_1}(k) < u_{rm}(k) < \frac{2\zeta\omega_n\varepsilon + x_{rm_2}(k+1)}{\omega_n^2} + x_{rm_1}(k).$$

Using the above inequality and the first two inequalities in equation (3.7), the state constraints are derived and depicted in the last two inequalities of equation (3.9). All the constraints included in Multi-Objective Optimization problem are collected together in equation (3.9) as a matrix compact form.

$$\begin{bmatrix} -C_2 \\ C_2 \\ -\phi \\ \phi \\ -C_2 \\ C_2 \end{bmatrix} \Delta U \leq \begin{bmatrix} -U_{min} + C_1 u_{rm}(k_i - 1) \\ U_{max} - C_1 u_{rm}(k_i - 1) \\ -Y_{min} + Fx(k_i) \\ Y_{max} - Fx(k_i) \\ \frac{-1}{\omega_n^2}\left(x_{rm_2}(k_i) + \omega_n^2 x_{rm_1}(k_i - 1) - \Delta_{lb}\right) + C_1 u_{rm}(k_i - 1) \\ \frac{1}{\omega_n^2}\left(x_{rm_2}(k_i) + \omega_n^2 x_{rm_1}(k_i - 1) + \Delta_{ub}\right) - C_1 u_{rm}(k_i - 1) \end{bmatrix}. \tag{3.9}$$

and $\Delta_{lb} = \Delta_{ub}$, where Δ_{ub} is an upper bound representing all the constants of the derivation bundled together. In the presented MPC architecture, constraints cannot be placed on the un-measured state. However, this is typically not an issue, since these constraints can be transformed into constraints on the control input. For example, the upper limit on the oscillation in the system results in an upper bound for the system states. Hence, using the state equations, a bound on the control inputs can be defined as shown above.

3.1.2 Chance-Constrained Model Predictive Control as a Reference Shaper

This section briefly introduces Chance-Constrained Model Predictive Control as a means to shape the reference command. The priority of both the control architectures shown in Figure 1.2 and 1.3, is to cancel the uncertainty and track the linear reference model dynamics of (1.4). The remaining control authority, which can be used by the CCMPC, can be used to shape the reference command u_{rm}. It is obtained by solving (1.5) subject to $u \in D_u$.

The discretized version of the reference model dynamics of (1.4) subject to a stochastic uncertainty is used as a model for the CCMPC:

$$x_{rm}(k+1) = A_{rm}x_{rm}(k) + B_{rm}u_{rm}(k) + B_{rm}w(k). \tag{3.10}$$

Here, (A_{rm}, B_{rm}) denote the discretized version of the original matrices, $x_{rm}(k)$ denotes the discrete-time state at the time instant k and $w(k) \in \mathbb{R}^m$ represents an integrated Wiener process. Let $\Delta u_{rm}(k) = u_{rm}(k) - u_{rm}(k-1)$ be a future control input, let $N_c > 0$ and $N_p \geq N_c$ denote the control and prediction horizon, as mentioned earlier. Let $\Delta U = [\Delta u_{rm}(k) \cdots \Delta u_{rm}(k+N_c-1)]^T$ with $\Delta U \in \mathbb{R}^{N_c \cdot m}$ be the incremental control sequence that captures the optimal trajectory and let $W = [w(k) \cdots w(k+N_p-1)]^T$ with $W \in \mathbb{R}^{N_p \cdot l}$. Denote $X \in \mathbb{R}^{N_p \cdot l}$ to be the vector of predicted states, then

$$X = G_{xx}x_0 + G_{xu}\Delta U + G_{xw}W, \tag{3.11}$$

where the matrices G_{xx}, G_{xu}, G_{xw} are generated by repetitive application of (3.10) which G_{xx} and G_{xu} corresponds to F and ϕ in (3.4), respectively. Considering the reference model dynamics in (1.4), the available control authority at the time instant k is then obtained by solving (1.5) for $u_{rm}(k)$ with $u \in D_u$.

Let F_X denote a convex polytope of state constraints. Consider a cost function $\mathscr{F}(\bar{X}, \Delta U)$, which is convex in $(\bar{X}, \Delta U)$. The CCMPC problem then is

$$\begin{gathered} \min_{\Delta U} \mathscr{F}(\bar{X}, \Delta U) \\ u \in D_u \\ P(X \notin F_X) \leq \delta \\ X = G_{xx}x_0 + G_{xu}\Delta U + G_{xw}W. \end{gathered} \tag{3.12}$$

Despite the linear dynamics in (3.10), the CCMPC problem is non-convex. In principle, the CCMPC problem can be solved with nonlinear solvers, although it needs to be relaxed to a convex approximation of the problem for practical implementation so that the nonlinear programming methods will be able to find the global optimum (see [68]). In order to relax the optimal control problem and obtain a solution, either nonlinear solvers or approximating methods can be employed (see e.g. [11, 69]). In any case, the first component of ΔU is applied to the reference model in (1.4). Note, that the noise vector $w(k)$ from (3.10) is only used for the CCMPC model and not implemented in the reference model dynamics of (1.4).

To solve the CCMPC problem here, we modify the output constraints by placing a normal distribution, $f(z \mid \mu, \sigma)$, on each one. The mean of the distribution, μ, is the previous limit for each constraint (X_{min}, X_{max}) and the standard deviation is an arbitrary small value, $\sigma \ll 1$. By using the inverse cdf (or quantile function) of δ in (3.12), the value by which the constraints are tightened, is found:

$$c = F^{-1}(\delta \mid \mu, \sigma). \tag{3.13}$$

Now, based on (3.13) and (3.7), the general MPC problem is solved with the updated constraints as below:

$$\begin{bmatrix} -C_2 \\ C_2 \\ -G_{xu} \\ G_{xu} \end{bmatrix} \Delta U \leq \begin{bmatrix} -U_{min} + C_1 u_{rm}(k-1) \\ U_{max} - C_1 u_{rm}(k-1) \\ -X_{min} + (c - X_{min}) + G_{xx}x(k) \\ X_{max} - (c - X_{max}) - G_{xx}x(k) \end{bmatrix}. \tag{3.14}$$

3.1.3 Model Reference Adaptive Control (MRAC)

In this section, we briefly review MRAC [27, 70, 71]. We begin first by describing model reference control architecture for systems without any uncertainty. Let $x(t) \in R^l$ be the state vector, let $u(t) \in R^m$ denote the control input and consider the following linear system without uncertainty.

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{3.15}$$

where $A \in R^{l\times l}, B \in R^{l\times m}$. We assume that the pair (A,B) is controllable and that B has full column rank. We assume $u(t)$ is restricted to the class of admissible control inputs consisting of measurable functions and $x(t)$ is available for full-state feedback.

A designer-chosen linear reference model is used to characterize the desired closed-loop response of the system,

$$\dot{x}_{rm}(t) = A_{rm}x_{rm}(t) + B_{rm}r(t). \tag{3.16}$$

The reference command $r \in R^m$ is assumed to be bounded and piecewise continuous, the matrix $A_{rm} \in R^{l\times l}$ is chosen to be Hurwitz and $B_{rm} \in R^{l\times m}$ is such that steady state accuracy is ensured. A nominal control law including a linear feedback part $u_{fb}(t) = K_m^T x(t)$ with $K_m \in R^{l\times m}$ and a linear feedforward part $u_{ff}(t) = K_b^T r(t)$ with $K_b \in R^{1\times m}$ is proposed to have the following form:

$$u(t) = u_{ff}(t) + u_{fb}(t). \tag{3.17}$$

Substituting (3.17) into (3.15) we have:

$$\dot{x}(t) = (A + BK_m^T)x(t) + BK_b^T r(t). \tag{3.18}$$

The design objective is to have (3.18) behave as the reference model in (3.16). To that effect, we introduce the following model matching conditions,

$$\begin{aligned} A + BK_m^T &= A_{rm}, \\ BK_b^T &= B_{rm}. \end{aligned} \tag{3.19}$$

assuming the model matching conditions exist. Defining the tracking error to be $e(t) = x(t) - x_{rm}(t)$ yields,

$$\dot{e}(t) = A_{rm}e(t). \tag{3.20}$$

which shows that the tracking error converges to zero exponentially fast since A_{rm} is Hurwitz. It follows from Lyapunov theory that there exists a unique positive-definite matrix $P \in R^{l\times l}$ satisfying the Lyapunov equation,

$$A_{rm}^T P + PA_{rm} + Q = 0. \tag{3.21}$$

for any positive-definite matrix $Q \in R^{l\times l}$.

MRAC with Uncertain Nonlinearity

Here we discuss how the effect of an additive matched uncertainty can be negated using MRAC. Suppose that we have an uncertain nonlinear model with a matched modeling uncertainty, Δ. Then equation (3.15) becomes,

$$\dot{x}(t) = Ax(t) + B(u+\Delta)(t). \tag{3.22}$$

Consider an adaptive term u_{ad} which needs to be designed to cancel out $\Delta(x)$. So, we modify our control law in equation (3.17), as

$$u(t) = u_{ff}(t) + u_{fb}(t) - u_{ad}(t). \tag{3.23}$$

With (3.23), (3.22) and (3.16), the tracking error dynamics would be:

$$\dot{e} = A_{rm}e + B(\Delta - u_{ad}). \tag{3.24}$$

Consider the well-known MRAC weight update law $\dot{W}(t) = -\Gamma_w \beta(x) e^T(t) PB$ that guarantees the tracking error and adaptive parameters are uniformly ultimately bounded, however, the parameters (W) stay bounded within a neighborhood of the ideal parameters (W^*) only if $\beta(x)$ is persistently excited [72]. Here, Γ_w denotes a positive-definite learning rate matrix, β is the known basis function for uncertainty parametrization, and P is the positive-definite matrix satisfying the Lyapunov equation. Chowdhary et al. in [35] used concurrent learning to overcome this problem. Furthermore, in [30] it was shown that when Gaussian Processes are used as adaptive elements, the architecture becomes capable of adapting to a wide class of uncertainties without having to pre-specify the structure of the adaptive elements. Here, we take advantage of GPs to learn the uncertainty in the model by modeling the adaptive element as the mean of an online-learned GP.

3.1.4 Gaussian Process MRAC

In the presented architecture, the stochastic uncertainties, Δ_i, in (1.1) are modeled each as a Gaussian Process (GP) [see (1.2)]. A GP is a distribution over functions and a sample drawn from a GP will be a function completely characterized by the mean and variance of the GP [28]. In GPs, the function is modeled as a linear combination of covariance kernel functions [28]. A common choice for defining the covariance, K, is the squared exponential function:

$$k(x,y) = exp(-\frac{\parallel x-y \parallel^2}{2\sigma^2}). \tag{3.25}$$

although the presented method can accommodate other kernels. The kernel function generates a mapping, ψ, to an infinite dimensional Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, such that k is an inner product defined by $k(x,y) = \langle \psi(x), \psi(y) \rangle_H$. In this RKHS, the mean, m, of the data generating stochastic process, Δ, is a linear combination of nonlinear bases:

$$m(x) = \sum_{j=1}^{\infty} \alpha_j k(x, x_j). \tag{3.26}$$

where $\alpha_i \in \mathcal{H}$ is a countably infinite dimensional vector, $k \in \mathcal{H}$ are the kernel functions evaluated over data points, x_j, vectors in the dual space. This rather general model can be learned directly from data, let $\mathcal{D} = \{(X_1, y_1)(X_2, y_2), \ldots, (X_n, y_n)\}$ be a set of state measurements sampled from an environment. Then assuming normal distribution as the prior on the data, the GP posterior mean and variance of the GP given the measurements are estimated as:

$$\begin{bmatrix} y_t \\ y_{t+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathcal{D},\mathcal{D}) + \omega^2 I & k(\mathcal{D},x) \\ k(\mathcal{D},x)^\top & k(x,x) \end{bmatrix}\right), \tag{3.27}$$

The posterior distribution is computed by conditioning the joint distribution over the new measurement, y_{t+1}, to obtain,

$$p(y_{t+1} \mid y_t, x_t, \mathcal{D}) \sim \mathcal{N}(\hat{m}(x_{t+1}), \hat{\Sigma}(x_{t+1})), \tag{3.28}$$

where,

$$\begin{gathered} \hat{m}(x_{t+1}) = \alpha^\top k(x_{t+1}, \mathcal{D}), \\ \hat{\Sigma}(x_{t+1}) = k(x_{t+1}, x_{t+1}) - k(\mathcal{D}, x_{t+1})^\top [K(\mathcal{D},\mathcal{D}) + \omega^2 I]^{-1} k(\mathcal{D}, x_{t+1}). \end{gathered} \tag{3.29}$$

are the predictive mean and covariance, respectively and $\alpha = K(\mathcal{D},\mathcal{D}) + \omega^2 I]^{-1}\bar{y}$. Hence, the mean is directly estimated from the set of available data. In essence, at any given point in time, an instantiation of the GP can be thought of as a Gaussian Radial Bases Function Network (RBFN). However, the main strength of the GP is that it does need to assume an a-priori allocation of RBF centers.

However, the main disadvantage of using the traditional GP regression techniques is that the covariance matrix increases in size as the size of $\mathcal{D}$ increases. In online applications, this can quickly become intractable as computing the inverse in (3.29) can become computationally intractable. It was shown in [30] that this problem can be alleviated using online sparsification techniques, and in particular Csato and Opper's budgeted online Sparse Gaussian Process regression technique [29] which only includes valuable data points in an active *Basis Vector set* ($\mathcal{BV}$). When new data is observed, the sparsification algorithm computes how well the new data point can be approximated by the existing basis vectors using a comparative test called the *kernel linear independence test* which $\forall \psi \in \mathcal{BV}$ is defined as:

$$\gamma_{t+1} = \| \sum_{i=1}^{t} \alpha_i \psi(x_i) - \psi(x_{t+1}) \|_{\mathcal{H}} . \tag{3.30}$$

The γ_{t+1} gives the residual distance between $\psi(x_i)$ and the GP generated by elements in $\mathcal{BV}$. An existing element, ψ_m, in the basis vector set which minimizes $D(\mathcal{GP} \parallel \mathcal{BV}) - D(\mathcal{GP} \parallel \mathcal{BV} \backslash \{\psi_m\})$ is removed and the new sample is added to the set. Given the basis vector set, the approximate mean and variance can be written from (3.29) as:

$$\begin{aligned} \hat{m}(x_{t+1}) &= \alpha^\top k(x_{t+1}, \mathcal{BV}), \\ \hat{\Sigma}(x_{t+1}) &= k(x_{t+1}, x_{t+1}) - k(\mathcal{BV}, x_{t+1})^\top [K(\mathcal{BV}, \mathcal{BV}) + \omega^2 I]^{-1} k(\mathcal{BV}, x_{t+1}). \end{aligned} \tag{3.31}$$

The reader is referred to [30] and [29] for details of the algorithmic implementation.

To achieve the tracking objective, the adaptive element is set to the online-learned mean of the stochastic process, Δ_i: $u_{ad}(x) = \hat{m}_i(x)$ at any state $x(t)$ from (1.6) to ensure that the tracking error goes to zero asymptotically.

$$\begin{aligned} u(t) &= u_{ff}(t) + u_{fb}(t) - u_{ad}(t), \\ u_{ad} &= \hat{m}(x). \end{aligned} \tag{3.32}$$

Algorithm 1 shows the steps necessary to be implemented for the presented MO-Op architecture.

Algorithm 1 GP-MRAC MO-Op MPC

```
Input: Initial states, reference command, $N_p$ and $N_c$
Initialize GP model from system states.
while reference command signal is available do
    Compute control objective function, $J$, by (3.5) and constraint matrices by (3.9).
    Optimize the objective function $J$ with (3.9) as constraint. Add multiple objectives as additional constraints.
    Set $u_{ad} = \hat{m}(x)$.
    Calculate $u_{fb} = K_m^\top x$ and $u_{ff} = K_b^\top u_{rm}$. $u_{rm}$ is the optimization output.
    Calculate control input using (1.5).
    Propagate system and reference model dynamics.
    Compute the nonlinear term of system dynamics.
    Update $\bar{x}_{rm}$ using (3.3).
end while
```

3.2 Adaptive Control under Switching Models

In this section we provide a method for modeling the nonlinear uncertainty, $\Delta(x)$, which is assumed to be a smooth deterministic function in the Hilbert space, $\mathcal{H}$. We look at a particular class of nonstationary switching functions $\bar{\Delta} = \{\Delta_1, \Delta_2, \ldots,\} \in \mathcal{H}$ that can be acted upon the system defined by (1.1). Our algorithm uses an adaptive control architecture which can learn the various models of uncertainties and detect change-points in the underlying model to actively cancel them out. GP-MRAC is already described in Section 3.1.4 and its online implementation is presented in [30].

3.2.1 Change-Point Detection with Gaussian Process Non-Bayesian Clustering

In most existing GP inference techniques, including the ones discussed in 3.1.4, the data is assumed to be generated from a single GP. However, this technique is not applicable to cases when the underlying data generating distribution could be switching in time. In particular, an algorithm that can determine the current generative distribution does not model well the data being observed is required when Δ_i is switching. In [31, 44], the GP-NBC method is proposed for identifying separable GP models from sequentially observable outputs, under mild assumptions [31], on the class of functions, $\mathcal{F}$, in which the various models exist. We present a brief overview of GP-NBC.

For a GP, the log-likelihood of a subset of points, y, can be evaluated as:

$$\begin{aligned}\log P(y \mid x, M) = &-\frac{1}{2}(y-\mu(x))^\top \Sigma_{xx}(y-\mu(x)) \\ &-\log|\Sigma_{xx}|^{1/2} + C.\end{aligned} \tag{3.33}$$

where $\mu(x) = K(X,x)^\top(K(X,X)+\omega_n^2 I)^{-1}Y$ is the mean prediction of the model, M, and $\Sigma_{xx} = K(x,x) + \omega_n^2 I - K(X,x)^\top(K(X,X)+\omega_n^2 I)^{-1}K(X,x)$ is the conditional variance plus the measurement noise. The log-likelihood contains two terms which account for the deviation of points from the mean, $\frac{1}{2}(y-\mu(x))^\top\Sigma_{xx}(y-\mu(x))$, as well as the relative certainty in the prediction of the mean at those points, $\log|\Sigma_{xx}|^{1/2}$. GP-NBC algorithm, at all times, maintains a set of points, S, which are considered unlikely to have arisen from the current GP model, M_c. The set S is used to create a new GP, M_S, which is tested against the existing models, M_i, using a non-Bayesian hypothesis test to determine whether the new model, M_S, merits instantiation as a new model, or should be clustered with an existing one. This test is defined as:

$$\frac{P(y \mid M_i)}{P(y \mid M_j)} \underset{\hat{M}_j}{\overset{\hat{M}_i}{\gtrless}} \eta. \tag{3.34}$$

where $\eta = (1-p)/p$, and $p = P(M_1)$. If the quantity on the left-hand side is greater than η, then the hypothesis M_i (i.e. that the data y is better represented by M_i) is chosen, and vice versa.

The GP-NBC algorithm is reproduced in Algorithm 2 (see [31, 44] for more details).

Algorithm 2 GP Clustering [31]

```
Input: Initial data $(X,Y)$, lps size $l$, model deviation $\eta$
Initialize GP Model 1 from $(X,Y)$.
Initialize set of least probable points $S = \emptyset$.
while new data is available do
    Denote the current model by $M_c$.
    If data is unlikely with respect to $M_c$, include it in $S$.
    if $|S| == l$ then
        for each model $M_i$ do
            Calculate log-likelihood of data points $S$ using having been generated from current model $M_i$ (3.33)
            $\log(S|M_i)$, and find highest likelihood model $M_h$, making $M_h$ current model.
            Create new GP $M_S$ from $S$.
            if $\frac{1}{l}(\log(S|M_S) - \log(S|M_c)) > \eta$ then
                Add $M_S$ as a new model.
            end if
        end for
    end if
end while
```

CHAPTER 4

Stability

4.1 Stability Analysis

This section discusses the stability of the presented adaptive-optimal control architecture. From the tracking error dynamics in (1.6), it follows that since A_{rm} is Hurwitz, if $||\Delta - u_{ad}||$ is bounded, the tracking error stays bounded. The main advantage of the reference command shaping architecture utilized here is that optimizing the output of the reference model using MPC will not change the tracking error dynamics in (1.6) [4].

Before considering the adaptive part of the control architecture, we first analyze the optimal part as well to check the stability. Since the chosen reference model is linear time-invariant and A_{rm} is a Hurwitz matrix the reference model is BIBO stable. The input which is the reference command, r, is bounded by design, hence the output of the optimal controller is also bounded and stable. Having multiple constraints does not make the system unstable as long as they are feasible. Nonetheless, the constraints tighten the bounds of the output. Thus, the output of the optimal part of the architecture enters the adaptive part bounded and stable.

Now, we investigate the error dynamics to show the stability of the entire architecture.

Remark 1 *The tracking error dynamics can be derived as follows:*

$$\begin{aligned}
\dot{e} = \dot{x} - \dot{x}_{rm} &= Ax + B(u+\Delta) - A_{rm}x_{rm} - B_{rm}u_{rm} \\
&= Ax + B(K_m x + K_b u_{rm} - u_{ad} + \Delta) - A_{rm}x_{rm} - B_{rm}u_{rm} \\
&= (A + BK_m)x - A_{rm}x_{rm} + BK_b u_{rm} - B_{rm}u_{rm} - Bu_{ad} + B\Delta \\
&= A_{rm}e + B(\Delta - u_{ad}).
\end{aligned}$$

Note that due to the matching conditions in (3.19), the reference signal is canceled out, hence, the stability of the architecture is not affected directly by the MPC-based reference command optimization. Furthermore, unlike previous learning-based MPC architectures [9, 27], it is not required to explicitly account for the effect of including MPC in the closed-loop in the stability analysis.

Let $\xi(t) \in \mathbb{R}^{n_2}$ be a Wiener process, $\sigma(t) \in \mathbb{N}$ denote a switching index that increments every time the set $\mathcal{BV}$ is modified, and G_σ be a linear operator associated to the covariance kernel $k(z,z')$ [28]. Following the analysis in [30] the uncertainty can be presented as,

$$\Delta(z) = m(z(t)) + G_\sigma(t,z(t))\, d\xi(t). \tag{4.1}$$

Hence, the tracking error dynamics (1.6) can be written as,

$$\dot{e} = A_{rm}e + B\varepsilon_m^\sigma + BG_\sigma\, d\xi. \tag{4.2}$$

where $\varepsilon_m^\sigma(z) = m(z) - \hat{m}^\sigma(z)$ and $u_{ad}(z) = \hat{m}^\sigma(z)$ is the estimation of the mean function $m(z)$. Time and state dependency of $G_\sigma(t,z(t))$ is dropped for the sake of brevity. Now to prove stability, we need to bound $\|\Delta(z) - u_{ad}(z)\|$. It is shown in [30] that there exists the following bound on $\varepsilon_m^\sigma(z)$:

$$\|\varepsilon_m^\sigma(z)\| \leq \frac{2\kappa^2 M^\sigma \sqrt{k_{max}}}{\omega^4} + \frac{\kappa k_{max} M^\sigma}{\omega^2}. \tag{4.3}$$

where k_{max} is the greatest kernel approximation error, κ is the maximum value the kernel can take (1 for Gaussian kernel), ω is the process noise, and M^σ is the largest value of an outlying measurement [30].

Using the above bound, the following theorem guarantees the stability of the presented architecture by proving the boundedness of tracking error:

Theorem 1 *Assume the system in (1.1), the control law in (1.5), and the uncertainty $\Delta(z)$ which is representable by a Gaussian Process* (1.2)*. Then, the outlined adaptive-optimal control Algorithm 1 and $u_{ad}(z) = \hat{m}^\sigma(z(t))$ guarantee that the system is mean square uniformly ultimately bounded a.s.*

Proof. The proof follows directly from [30]. The Itô differential generator $\mathscr{L}$ for a smooth function is defined in [30]. Let $V(e(t)) = \frac{1}{2}e^T(t)Pe(t)$ be the Lyapunov candidate, where P is the positive definite solution to the Lyapunov equation (3.21). The Lyapunov candidate is bounded below and above by $\frac{1}{2}\lambda_{min}(P)\|e\|^2 \leq V(e) \leq \frac{1}{2}\lambda_{max}(P)\|e\|^2$. The Itô differential of the Lyapunov candidate considering the solution of (4.2) for the σ^{th} system is:

$$\begin{aligned}\mathscr{L}V(e) &= \textstyle\sum_i \frac{\partial V(e)}{\partial e_i}Ae_i + \frac{1}{2}\sum_{i,j}[BG_\sigma(BG_\sigma)^T]_{ij}\frac{\partial^2 V(e)}{\partial e_i \partial e_j} \\ &= -\tfrac{1}{2}e^TQe + e^TPB[\varepsilon_m^\sigma(z) + G_\sigma\, d\xi(t)] + \tfrac{1}{2}\mathrm{tr}(BG_\sigma(BG_\sigma)^TP).\end{aligned} \tag{4.4}$$

Let $c_1 = \frac{1}{2}\|P\|\|BG_\sigma\|^2$ and $c_2 = \|PB\|$, then,

$$\begin{aligned}\mathscr{L}V(e) &\leq -\tfrac{1}{2}\lambda_{min}(Q)\|e\|^2 + c_2\|e\|\|\varepsilon_m^\sigma(z) + G_\sigma\, d\xi(t)\| + c_1 \\ &\leq -\tfrac{1}{2}\lambda_{min}(Q)\|e\|^2 + c_2\|e\|(\|\varepsilon_m^\sigma(z)\| + c') + c_1.\end{aligned} \tag{4.5}$$

From the mentioned bound in (4.3) and $c_3 = \frac{2\kappa^2}{\omega^4}$, $c_4 = \frac{\kappa}{\omega^2}$, we have $\|\varepsilon_m^\sigma(z)\| \leq c_3 M^\sigma \sqrt{k_{max}} + c_4 M^\sigma k_{max}$. So,

$$\mathscr{L}V(e) \leq -\frac{1}{2}\lambda_{min}(Q)\|e\|^2 + c_2\|e\|\left(c_3 M^\sigma \sqrt{k_{max}} + c_4 M^\sigma k_{max} + c'\right) + c_1. \tag{4.6}$$

It is also shown in [30] that $\|G_\sigma d\xi\| \leq c'$.

Define $c_5 = c_3\sqrt{k_{\max}} + c_4 k_{\max}$. Therefore, outside of the set,

$$\Theta_\gamma^\sigma = \left\{ \|e\| \geq \frac{c_5 M^\sigma + \sqrt{(c_5 M^\sigma)^2 + 2\lambda_{min}(Q)c_1}}{\lambda_{min}(Q)} \right\}. \tag{4.7}$$

■

$\mathscr{L}V(e) \leq 0$ a.s.

Now it is required to prove the existence of M^σ. To this effect it is sufficient if we show the set Θ_γ^σ is bounded. As it is mentioned earlier, the reference model is BIBO stable. Therefore, the only way for Θ_γ^σ to go unbounded is if the bound in (4.3) does not exist which implies that $\hat{m}^\sigma(z)$ cannot estimate the mean function $m(z)$. Based on the GP-MRAC algorithm outlined, it is guaranteed that the ideal weights for RBFN representation of $m(z)$ can be computed. Thus, if $\|\varepsilon_m^\sigma(z)\|$ is unbounded, it results in violating the universal approximation theorem for RBFs which holds for RBFNs with Gaussian kernels [73]. This leads to contradiction, showing $\|\varepsilon_m^\sigma(z)\|$ and Θ_γ^σ are bounded and hence M^σ exists.

Remark 2 *The bound in* (4.7) *gets tighter as the kernel density increases. The ability to handle higher kernel density directly depends on the available on-board processing power and memory, hence, increasing processing power and memory can help reduce the error bound. The bound in* (4.7) *gets looser with large outliers M^σ, but can be balanced with larger regularizing parameter ω. Finally, note that unlike RBFN-based proofs [74], operation over a compact domain where kernels have been a-priori allocated need not be assumed, the domain can grow with data, but the kernel budget restricts attainable accuracy.*

Remark 3 *The above theorem guarantees that the system states will be driven inside a positively invariant set exponentially fast. Therefore if the switching interval between any two consecutive model (Δ_i) switches is sufficiently large, the stability of the switching nonlinear dynamical system in* (1.1) *can be guaranteed. Note that the required time interval between any two switches will always be finite, because of the exponential convergence guarantees.*

We now analyze the effect on the stability due to potential miss-classification by GP-NBC of the underlying model. Note first that GP-NBC is guaranteed to correctly classify the underlying model within a finite number of samples [31]. This property allows us to prove the following corollary:

Corollary 1 *Consider a system and the tracking error dynamics as described by* (1.1) *and* (1.6). *If the worst case ε_m is defined as $\varepsilon_w = \mid m - \hat{m}_{worst} \mid$ where $\hat{m}_{worst} = \sup_{i \in \Upsilon}(\hat{m}_{true} - \hat{m}_i)$ and the probability of making an incorrect prediction and detecting a switch point are defined as δ_L and δ_D, respectively, then, the growth in the tracking error due to a potential miss-classification by GP-NBC is bounded with probability* $1 - \delta_L - \delta_D$.

Proof. The presented proof is based on the worst case ε_w which is calculated using the largest difference of ε and follows the analysis used in [31] to obtain an upper bound on the growth of the tracking error. By applying the triangle inequality to the solution of the tracking error dynamics (1.6), we obtain the bounds on $e(t)$ as,

$$\parallel e(t) \parallel \leq \parallel exp(A(t-t_0))e(t_0) \parallel + \parallel \int_{\tau}^{t} exp(A(t-\tau))B(\tau)(\Delta - u_{ad})d\tau \parallel \tag{4.8}$$

From [31] there exists an upper bound on the total number of time steps GP-NBC can make an incorrect prediction when the model has switched. This guarantees that the integral defined in the transient response is finite. Since $\Delta - u_{ad}$ is upper bounded by ε_w the above equation reduces to

$$\parallel e(t) \parallel \leq \parallel exp(A(t-t_0))e(t_0) \parallel + \parallel A^{-1}[exp(A(t)) - I]B(t)\varepsilon_w \parallel \tag{4.9}$$

Since the total number of time steps GP-NBC makes an incorrect prediction is $n + (m-1)\mathcal{N}_c(U_E, d_x)$ with probability $1 - \delta_L - \delta_D$ [31], the integral in the transient response is finite with the same probability. Here n is the worst case bound on the number of samples such that GP makes an incorrect prediction, m is the window size to sufficiently distinguish any two models and $\mathcal{N}_c(U_E, d_x)$ is the covering number [31]. ■

Remark 4 *Corollary 1 guarantees that the error during the transition between models will remain bounded in probability.*

CHAPTER 5

Results

5.1 MO-Op MPC Results

In this section we evaluate the presented architecture (Figure 1.2) through simulation on a representative flight control problem. We consider the wing-rock dynamics problem [75]. However, we add to the problem an additive emulated rigid-flexible mode oscillatory term: $p^2 sin(p)$. This term oscillates like the solution to Van der Pol equation with resonance terms, which is an oscillator with nonlinear damping. The frequency of oscillation is modeled to increase with the roll rate p, while the amplitude is modeled to scale with the square of the roll rate. Consider the uncertain nonlinear dynamical model of wing-rock dynamics with the additive term (1.1):

$$\begin{bmatrix} \dot{\phi} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \delta_a + \Delta + p^2 \sin(p). \tag{5.1}$$

where $\delta_a = u$ in (5.1) is the aileron deflection, ϕ is the roll angle, and p is the roll rate. The generic nonlinear and time-varying uncertainty, Δ, takes on the following form for wing-rock dynamics: $\Delta = W_0 + W_1\phi + W_2 p + W_3|\phi|p + W_4|p|p + W_5\phi^3$. It is assumed that the oscillatory term has frequencies that are beyond the bandwidth of the actuator system. Therefore, it cannot directly be canceled using constrained control input. Therefore, finding a control input that minimizes the oscillation in the system forms the objective in our MO-Op problem alongside the tracking performance, which is our first objective.

The natural frequency and the damping ratio of the chosen reference model (second-order system) are $\omega_{rm} = 2$ and $\zeta_{rm} = 0.5$, so that:

$$\begin{bmatrix} \dot{x}_{rm_1} \\ \dot{x}_{rm_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} \begin{bmatrix} x_{rm_1} \\ x_{rm_2} \end{bmatrix} + \begin{bmatrix} 0 \\ 4 \end{bmatrix} u_{rm}. \tag{5.2}$$

The feedback and feedforward gains are $K_m = [-4 \;\; -2]$, $K_b = 4$, respectively. The simulation runs for 50 seconds with a time step of 0.01 second.

First, we present the results for the GP-MRAC-MPC without the oscillation term. The only objective function here is the tracking error. The results are shown in Figure 5.1. It can be seen that the GP-MRAC-MPC architecture can achieve good tracking performance.

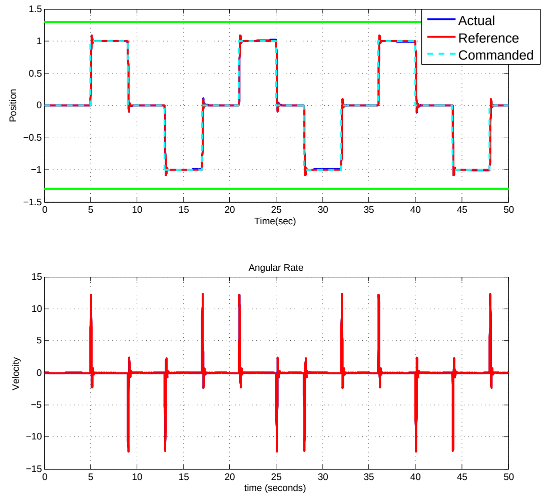


Figure 5.1: Position and velocity response for GP-MRAC-MPC architecture without the oscillation term.

Now we present results with the oscillatory term added. The oscillation term cannot be directly canceled by GP-MRAC, since the dynamics tend to get increasingly aggressive as p increases and the control input given by the actuators is practically limited. Even in simulation, GP is not able to overcome the complexity of this oscillation term as shown in Figure 5.2.

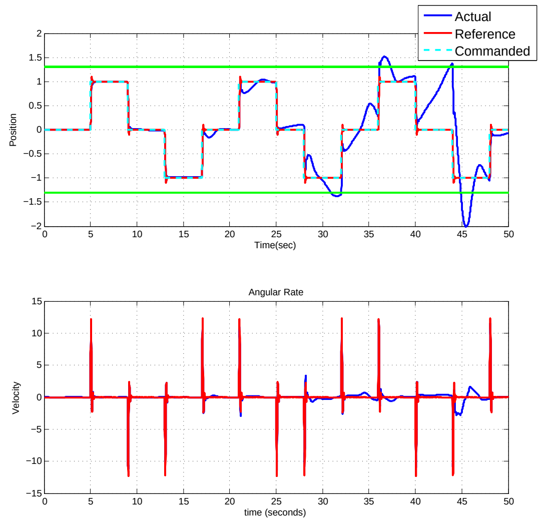
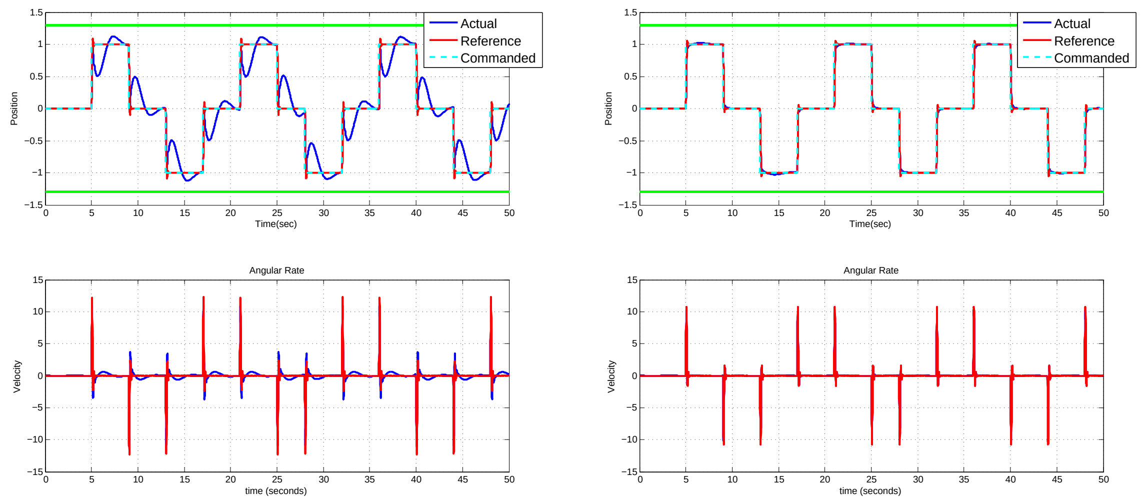


Figure 5.2: Position and velocity response by GP-MRAC learning of the oscillation term.

The left plot in Figure 5.3 shows the performance of the baseline GP-MRAC-MPC controller without

MO-Op. The results show that the oscillatory term results in sub-optimal performance. To handle this issue in the MO-Op framework, the second objective is posed as a constraint in our MPC problem, utilizing the ε-constraint MO-Op method. Since the energy in the system is a measure of oscillation, for minimizing the oscillation, the root mean square of the roll rate, P_{rms}, can be computed and a control strategy can be sought that restricts the oscillation below a user-specified upper bound. The bound on P_{rms} results in an upper bound for roll rate itself. The exact value of this bound was found by plotting the maximum P_{rms} versus upper bound. The plot for the entire set of simulations that is run for different upper bounds, has two local minima while one of them has the better tracking performance. The smaller upper bound does not result in the best tracking performance, since it causes the system to discard some frequencies that are needed. Utilizing that fact about the better the tracking performance is, the more desired the result is, an upper bound on the roll rate can be found. After evaluating that upper bound, the MO-Op MPC problem can be formed in ε-constraint manner by utilizing that upper bound as a constraint on p as explained in section 3.1.3.1.1. As discussed in section 2.1, ε-constraint method is selected as the best choice for online flight control problem since it is more amenable to online computation. The specific disadvantages of other methods were discussed in Section 2.1. The results for the solution to the MO-Op MPC problem, compared to the baseline with tracking error as the only objective, are shown in Figure 5.3. The constraints on the system output shown in this figure are not real physical constraints but we further decreased them to be shown in the plot and also to reflect the controller ability to deal with tight constraints.



(a) Performance without MO-Op

(b) Performance with MO-Op

Figure 5.3: Comparison in performance without and with optimizing over the oscillation.

It can be seen that the MO-Op technique outperforms the baseline since the MO-Op method takes into account the potential for exciting the oscillatory mode. MO-Op formulation of the problem considers this potential by constraining the control input to a region where the oscillations are expected to be below the user-defined threshold. To show the whole range of velocity we are forced to have a wide range for its axis. Thus, the difference in velocity plots are not emphasized but it is evident that the roll rate becomes more compact due to the second objective bound.

By changing Δ_{ub} and storing P_{rms}, we investigated whether the performance of the controller, with respect to the second objective, is as expected. As shown in Table 5.1, minimum of the maximum allowed P_{rms} occurs at $\Delta_{ub} = 203$ while the plot in Figure 5.3(b), sets $\Delta_{ub} = 179$. This is as expected, since tracking is our first priority objective and the tracking performance related to $\Delta_{ub} = 179$ is the most desired one amongst the others [see Figure 5.6 and 5.7]. Therefore, although it did not acquire the least possible P_{rms}, which means the least oscillatory behavior, it obtains the best tracking performance with a slightly higher P_{rms}. That is what we anticipate from a MO-Op formulation.

Table 5.1: Maximum allowed P_{rms} vs. Δ_{ub}

P_{rms}	1.4623	1.4348	1.4278	1.4222	1.4221	1.4224	1.4232	1.4259	1.4480
Δ_{ub}	160	170	179	200	203	204	210	220	250

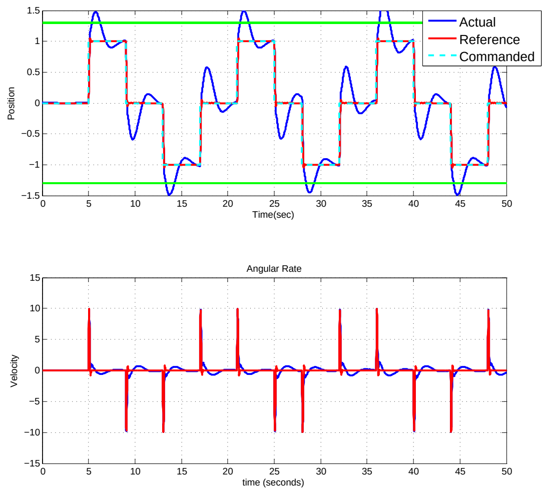
There are some parameters involved in the simulation, which are chosen by the designer. These parameters include Δ_{ub}, control and prediction horizon, and GP tuning parameters. The values of these parameters for the previous plots are shown in Table 5.2. In order to find the ones influencing the results, we report some comparisons by changing one parameter at a time in the following plots.

Table 5.2: Designer-involved parameters in the simulation

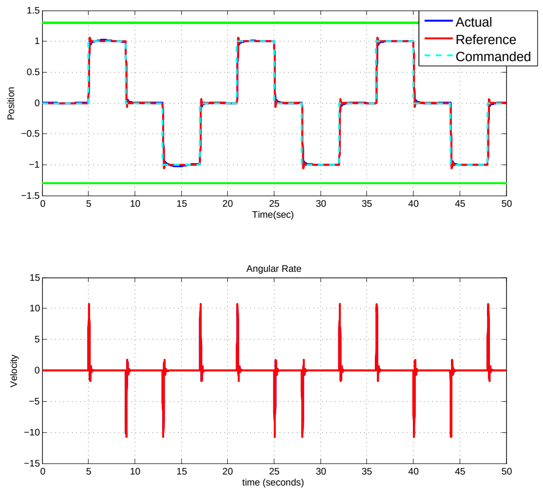
N_p	N_c	Δ_{ub}	σ	noise	n	tol
18	10	179	1.5	0.001	150	10^{-4}

The first set of results specifies changing the control horizon while the other parameters are selected as in Table 5.2.

Smaller control horizon will make a future control trajectory with less number of parameters, which makes the MPC controller less flexible as shown in Figure 5.4(a). On the other hand, making it equal to the prediction horizon, causes the controller to create control input more than its optimization window permits. Therefore, having a control horizon equal to prediction horizon usually results in not having the best performance as shown in Figure 5.4(b). If we choose $N_p = N_c = 20$, the difference will be more evident but here we refused to change more than a parameter at a time to emphasize each parameter's influence on the final



(a) Performance with $N_c = 2$

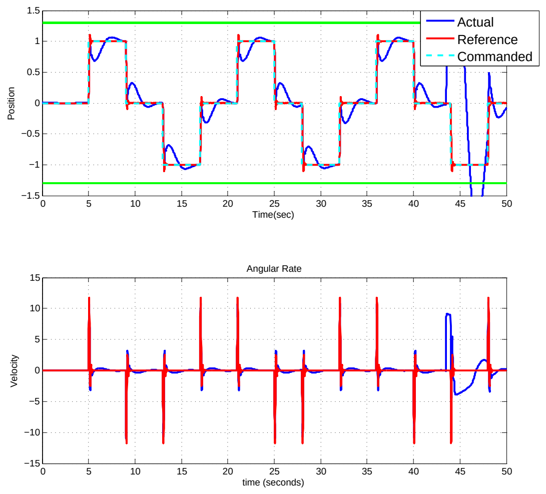


(b) Performance with $N_c = 18$

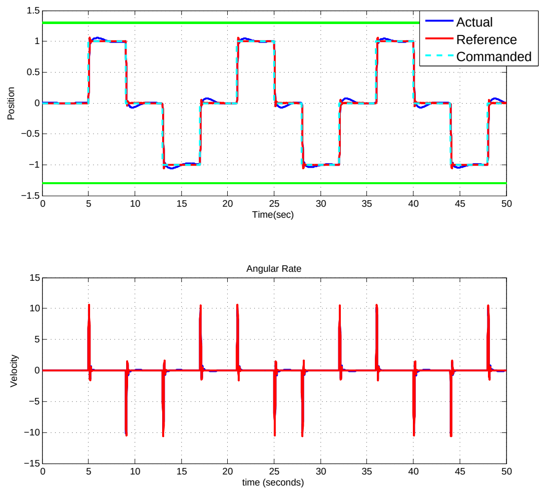
Figure 5.4: Comparison in performance while changing the control horizon.

result.

The second set of results shows the change in performance while we were changing the prediction horizon.



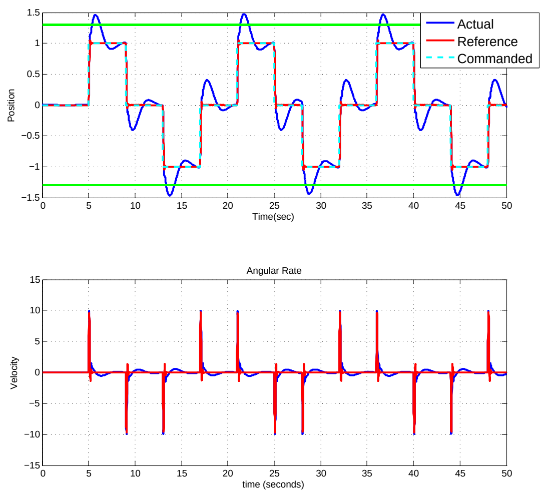
(a) Performance with $N_p = 10$



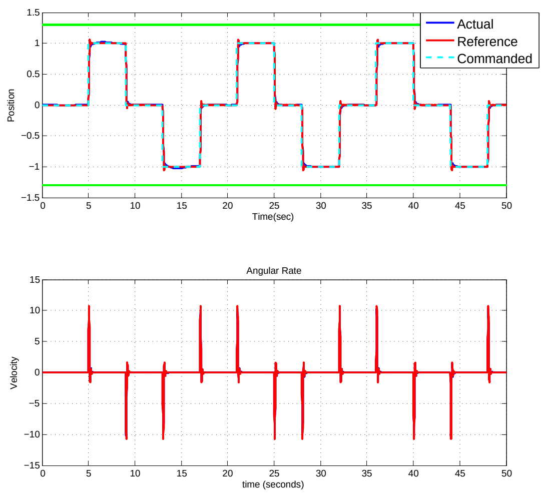
(b) Performance with $N_p = 20$

Figure 5.5: Comparison in performance while changing the prediction horizon.

As explained above, making N_p and N_c equally deteriorates MPC controller performance. In addition to that, reducing the optimization window size decreases the controller ability to predict the future as shown in Figure 5.5(a). Increasing the difference between control and prediction horizon too much will also result in a



(a) Performance with $\Delta_{ub} = 140$

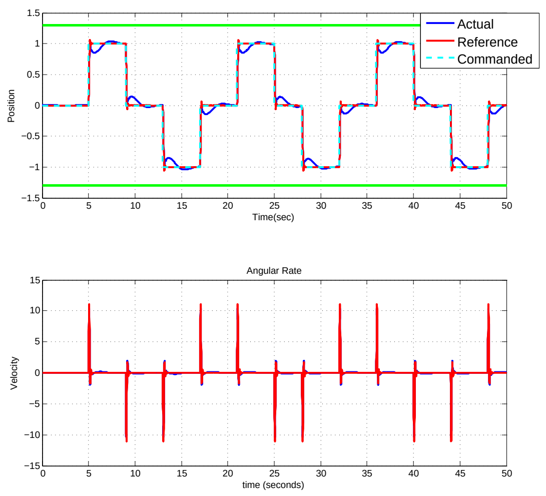


(b) Performance with $\Delta_{ub} = 179$

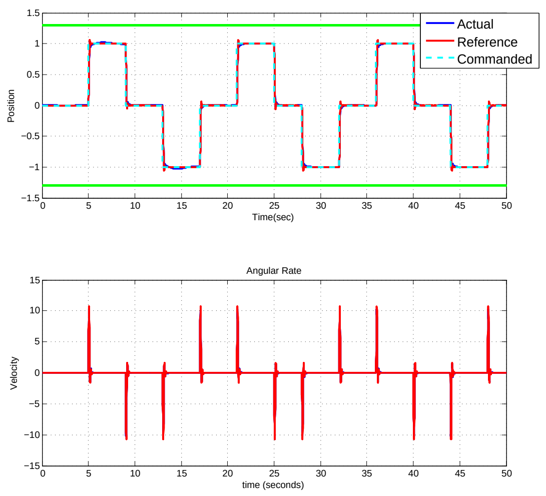
Figure 5.6: Comparison in performance while changing the upper bound.

less optimal behavior, demonstrated in Figure 5.5(b).

Figure 5.6 and 5.7 show the performance of the system based on the different upper bounds that the user chooses for the second objective.



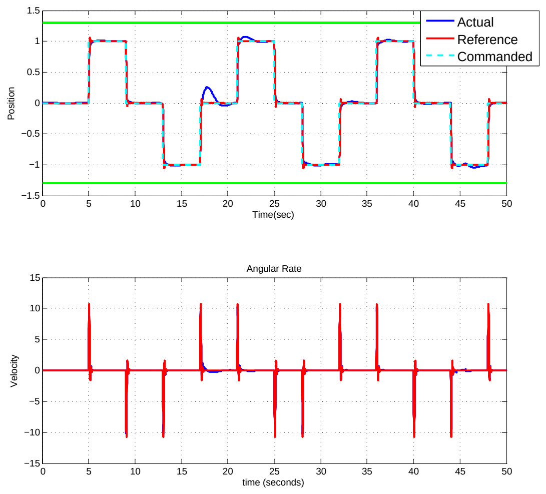
(a) Performance with $\Delta_{ub} = 200$



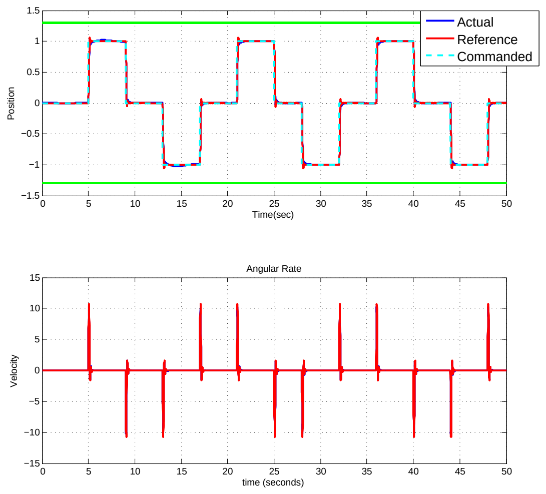
(b) Performance with $\Delta_{ub} = 179$

Figure 5.7: Comparison in performance while changing the upper bound.

Figure 5.8, 5.9, 5.10 and 5.11 demonstrate the result of the change in GP hyperparameters, σ, noise, number of centers, and tolerance, respectively. In all the upcoming figures, the right plots are for the values used by GP in MO-Op simulation shown in Figure 5.3(b).



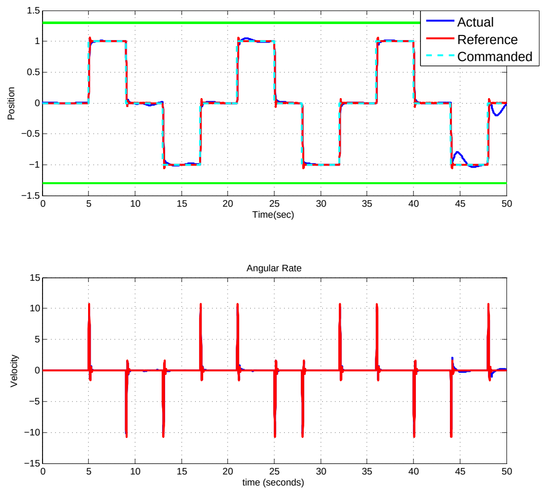
(a) Performance with $\sigma = 5.5$



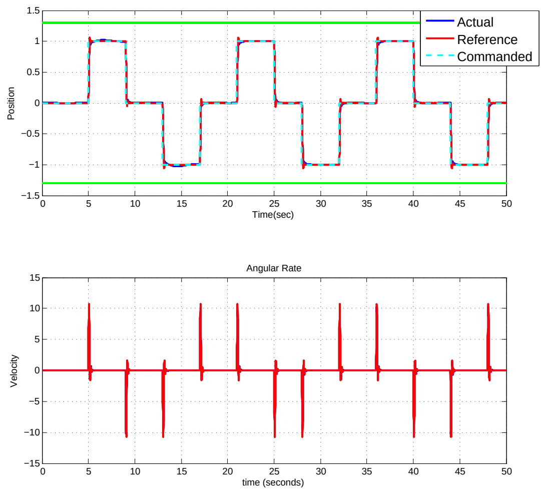
(b) Performance with $\sigma = 1.5$

Figure 5.8: Comparison in performance while changing GP hyperparameter, σ.

All the results for changing the GP hyperparameters suggest that the best regression performance by GP occurs at a proper adjustment for each parameter. While it is possible to have a satisfactory performance in the system for some changes in the parameters, the most optimal one occurs at a precise choice of these hyperparameters.



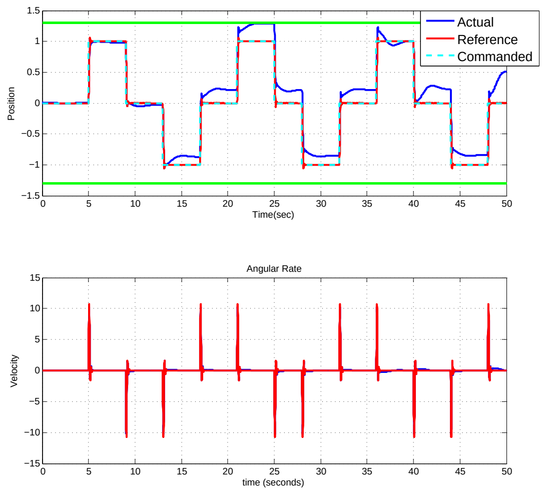
(a) Performance with noise $= 0.000001$



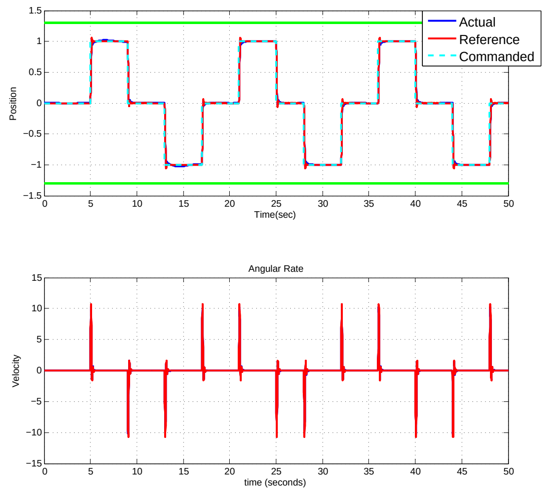
(b) Performance with noise $= 0.001$

Figure 5.9: Comparison in performance while changing GP hyperparameter, noise.

As it was also expected, the results are mostly influenced by σ and the number of centers, which play an important role in the way regression is done by GP.

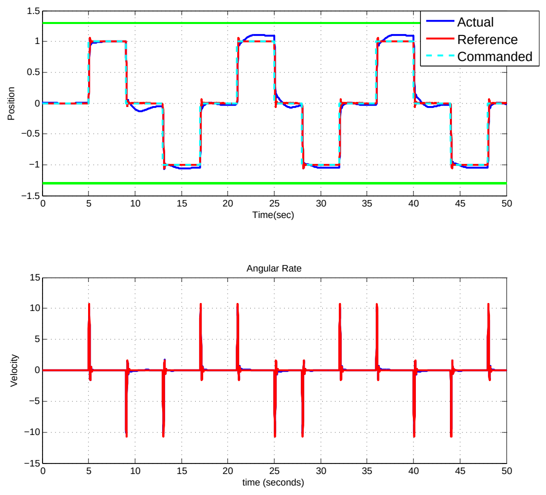


(a) Performance with $n = 10$

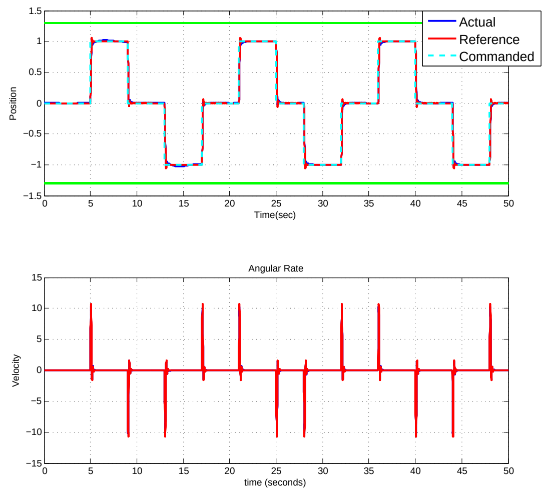


(b) Performance with $n = 150$

Figure 5.10: Comparison in performance while changing GP hyperparameter, number of centers.



(a) Performance with tol $= 1$



(b) Performance with tol $= 10^{-4}$

Figure 5.11: Comparison in performance while changing GP hyperparameter, tolerance.

The plots shown for simulation parameters comparison are sensitivity plots for the control architecture at hand. Except Δ_{ub} that is exclusive to this especial formulation of MO-Op problem, the others are generalizable to any other problem using similar parameters. GP hyperparameters sensitivity analyses are actually transferable to any regression problem. Thus, all these plots are validation of the simulation parameters' impact on this control architecture.

5.2 Adaptive-Optimal Clustering Results

Now we evaluate the adaptive-optimal clustering extension of the architecture through simulation on a representative flight control problem. Consider the uncertain nonlinear dynamical system in the form (1.1):

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (u+\Delta) \tag{5.3}$$

This class of models is applicable to the class of wing-rock dynamics:

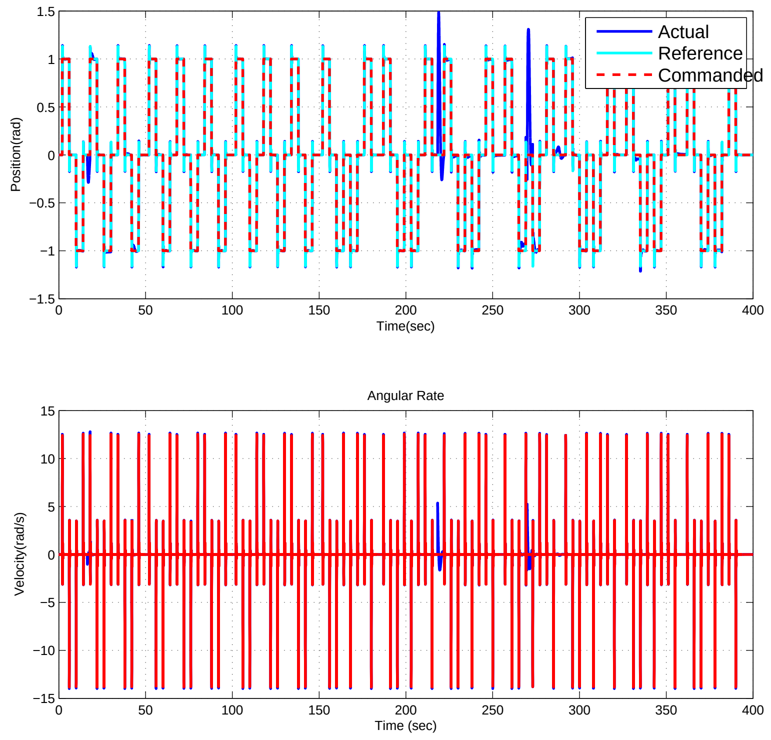
$$\begin{bmatrix} \dot{\phi} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (\delta_a+\Delta) \tag{5.4}$$

where $\delta_a = u$ in (5.1) is the aileron deflection, ϕ is the roll angle, and p is the roll rate. The generic nonlinear and time-varying uncertainty Δ takes on the following form for wing-rock dynamics $\Delta = W_0 + W_1\phi + W_2 p + W_3|\phi|p + W_4|p|p + W_5\phi^3$. Here we assume that the weights W are randomly switching between the three different sets of weights given below:

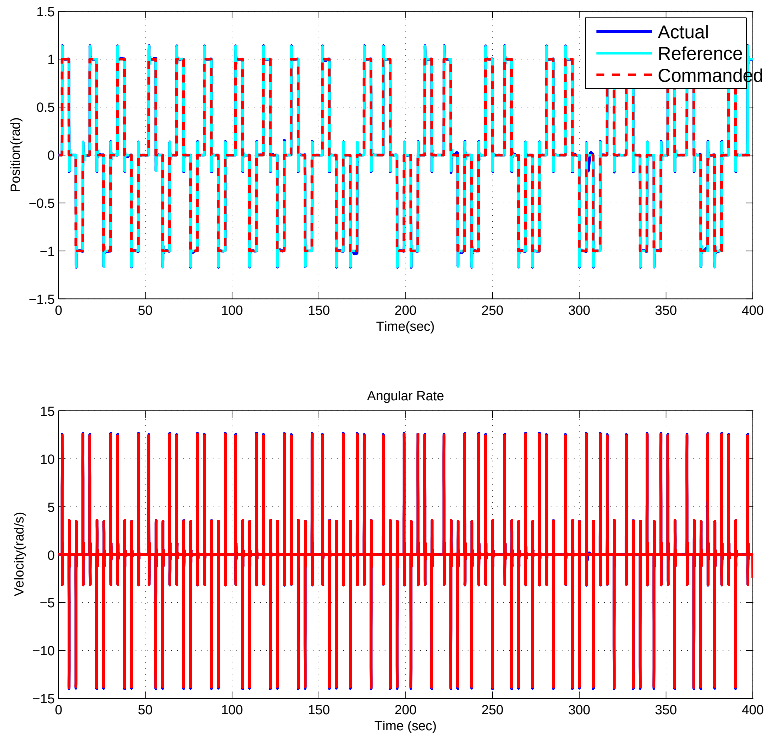
$$\begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \\ W_4 \\ W_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 0.2314 \\ 0.1918 \\ -0.6245 \\ 0.0095 \\ 0.0214 \end{bmatrix}, \begin{bmatrix} -3 \\ -0.01859 \\ 0.01516 \\ -0.06245 \\ 0.0095 \\ 0.0214 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.324 \\ -0.01516 \\ -0.06245 \\ 0.0095 \\ 0.0214 \end{bmatrix}$$

The reference model is the same as (5.2). The feedback and feedforward gains are $K_m = [-4 \quad -2]$, $K_b = 4$, respectively. The simulation runs for 250 seconds with a time step of 0.01 seconds.

The weights are assumed to switch in a manner unknown to the controller, however, we assume that there is a finite time interval between any two switches. In order to test the performance of our method, we compare the presented GP-MRAC-NBC algorithm with a GP-MRAC architecture which learns the uncertainty model without clustering. We show comparisons between the positions and velocities between both the methods along with the error performances.



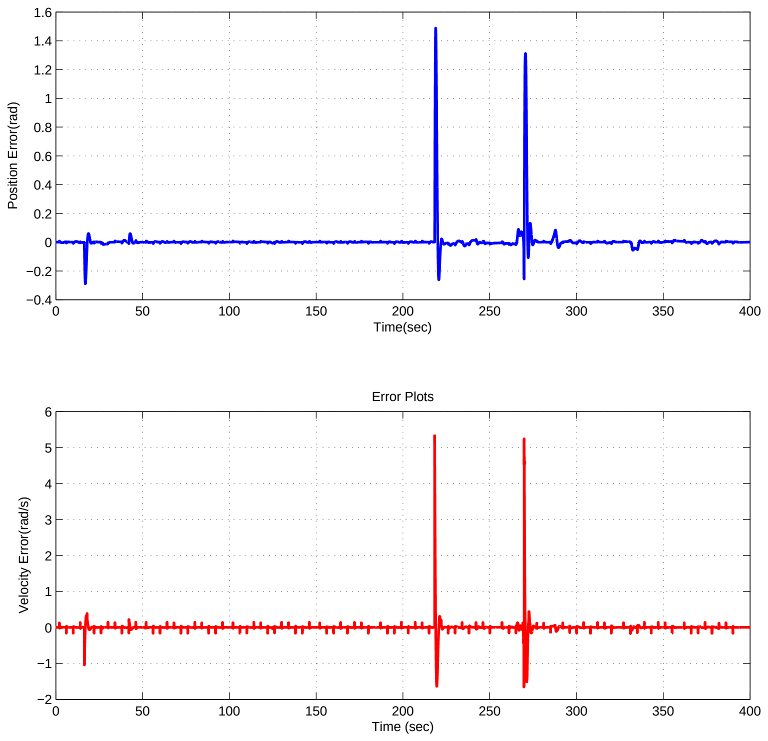
(a) Baseline Performance



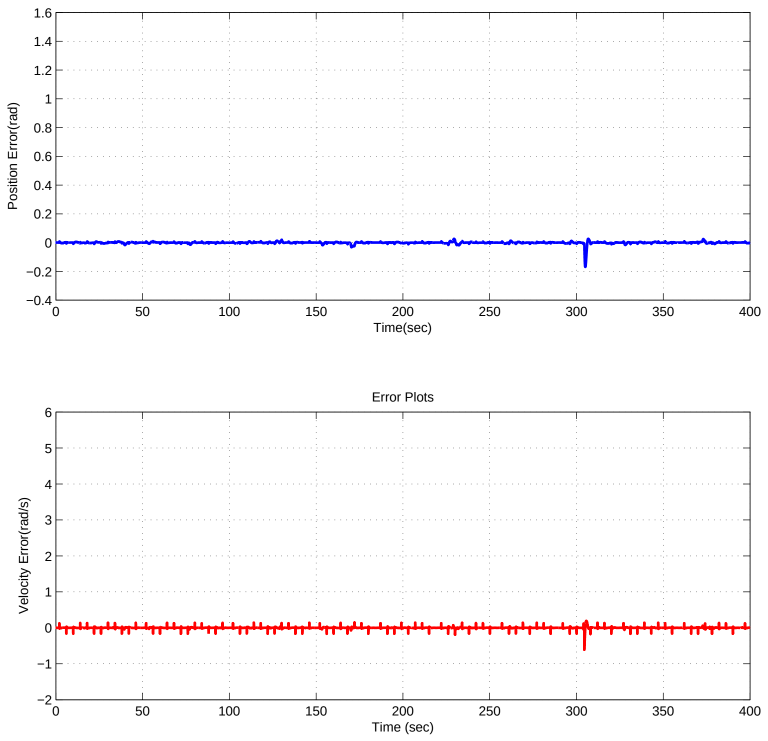
(b) Performance with Clustering

Figure 5.12: Comparison in performance without and with clustering the uncertainty models.

Figure 5.12 presents the tracking performance. The baseline GP-MRAC has suboptimal transient performance, since it can be seen that once the modeling error is switched to the second set of weights, the positions are not optimally tracked. However, in steady state, when GP-MRAC does learn the new model, GP-MRAC's performance is similar to GP-MRAC-NBC.



(a) Baseline Error



(b) Error with Clustering

Figure 5.13: Comparison in error performance without and with clustering the uncertainty models.

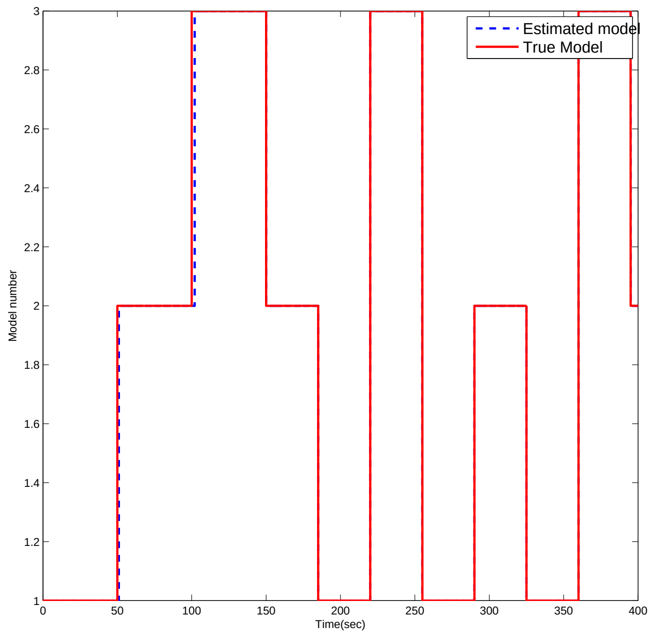


Figure 5.14: Model estimation performance of the Clustering Algorithm

Figure 5.13 compares the error performance of the baseline with the GP-MRAC-NBC. A degradation is noticed in the error performance when no clustering is performed. Also, it can be noticed that the performance of the clustering algorithm improves over time as the algorithm sees more and more data from a particular model. For example, the error of the system with the first set of weights has improved in performance when the first model is encountered the second time. The same is true for the second and third sets of weights. This indicates long-term learning and reuse of previous experience. Sharp peaks in the error plot indicate the points where the clustering algorithm is using the data to identify the right model and, hence, there is a small degradation in the performance; however, once identification is done, the tracking error significantly reduces. Figure 5.14 shows that the GP-NBC algorithm is able to correctly detect changes in the model and cluster together data from the same underlying GP.

CHAPTER 6

Conclusion

6.1 Conclusion

We presented a control architecture for Multiple-Objective Optimization (MO-Op) flight control. We demonstrated that the MO-Op flight control problem can be cast as a constrained learning-based adaptive Model Predictive Control (MPC) problem. The learning is performed using Gaussian Process adaptive elements, while the optimization is made online-realizable by relaxing some of the objectives posing them as state constraints. The architecture was validated on a nonlinear roll dynamics system with wing-rock effects and emulated state-dependent rigid-flexible mode interaction. The first objective was to minimize the tracking error, and the second one was to minimize the oscillation in the system in order to improve ride comfort. The stability of key elements of the presented adaptive-optimal architecture is discussed. We expect that the online feasibility and stability guarantees of the presented architecture will make it applicable to many other Multi-Objective Optimization flight control problems.

We also presented a GP-based adaptive-optimal control architecture for adaptive control of aerospace systems with dynamically changing stochastic modeling uncertainty. The key salient feature of our architecture is that not only can it detect changes, but it uses online GP clustering to enable the controller to utilize past learning of similar models to significantly reduce learning transients. The stability of the architecture was argued theoretically, and empirical results showed that it can be used in online settings. Results indicate the possibility of designing flight control and decision-making systems that can adapt to unforeseen situations; Moreover, it is able to leverage past experience to minimize sub-optimal performance by reducing the time spent in learning.

References

[1] Nhan Nguyen and James Urnes. Aeroelastic modeling of elastically shaped aircraft concept via wing shaping control for drag reduction. In *AIAA Atmospheric Flight Mechanics Conference*, page 1316, 2012.

[2] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer, 2008.

[3] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I–II. Athena Scientific, PO Box 391, Belmont, MA 02178, 2007.

[4] Maximilian Mühlegg, Girish Chowdhary, and Florian Holzapfel. Optimizing reference commands for concurrent learning adaptive-optimal control of uncertain dynamical systems. In *GNC*. AIAA, 2013.

[5] Ali Abdollahi, Rakshit Allamaraju, and Girish Chowdhary. Adaptive-optimal control of nonstationary dynamical systems. In *Proceedings of the 2015 European Guidance, Navigation, and Control Conference*, 2015. accepted.

[6] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In A. Garulli and A. Tesi, editors, *Robustness in identification and control*, volume 245 of *Lecture Notes in Control and Information Sciences*, page 207226. Springer Berlin / Heidelberg, 1999. 10.1007/BFb0109870.

[7] G. D. Nicolao, L. Magi, and R. Scattolini. *Nonlinear Model Predictive Control*, volume 26 of *Progress in Systems and Control Theory*, chapter Stability and Robustness of Nonlinear Receding Horizon Control, pages 3–22. Birkhäuser, Basel-Boston-Berlin, 2000.

[8] E. C. Kerrigan. *Robust Constraint Satisfaction Invariant Sets and Predictive Control*. PhD thesis, University of Cambridge, Department of Engineering, November 2000.

[9] Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[10] M. Ono. Closed-loop chance-constrained mpc with probabilistic resolvability. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 2611–2618, Dec 2012.

[11] Lars Blackmore and Masahiro Ono. Convex chance constrained predictive control without sampling. In *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2009.

[12] Kumpati S. Narendra and Anuradha M. Annaswamy. *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, 1989.

[13] Karl Johan Aström and Björn Wittenmark. *Adaptive Control*. Addison-Weseley, Readings, 2nd edition, 1995.

[14] Gang Tao. *Adaptive Control Design and Analysis*. Wiley, New York, 2003.

[15] S. Di Cairano and I.V. Kolmanovsky. Further developments and applications of network reference governor for constrained systems. In *American Control Conference (ACC), 2012*, pages 3907 –3912, june 2012.

[16] A. Bemporad, A. Casavola, and E. Mosca. Nonlinear control of constrained linear systems via predictive reference management. *Automatic Control, IEEE Transactions on*, 42(3):340 –349, mar 1997.

[17] E.G. Gilbert and I. Kolmanovsky. A generalized reference governor for nonlinear systems. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 5, pages 4222 –4227 vol.5, 2001.

[18] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Dover Publications, 2013.

[19] Veronica Adetola, Darryl DeHaan, and Martin Guay. Adaptive model predictive control for constrained nonlinear systems. *Systems & Control Letters*, 58(5):320–326, 2009.

[20] Hiroaki Fukushima, Tae-Hyoung Kim, and Toshiharu Sugie. Adaptive model predictive control for a class of constrained linear systems based on the comparison model. *Automatica*, 43(2):301–308, 2007.

[21] David Mayne. Nonlinear model predictive control: Challenges and opportunities. In *Nonlinear model predictive control*, pages 23–44. Springer, 2000.

[22] Suresh Kannan. *Adaptive Control of Systems in Cascade with Saturation*. PhD thesis, Georgia Institute of Technology, Atlanta Ga, 2005.

[23] Eugene Lavretsky. Combined/composite model reference adaptive control. *IEEE Transactions on Automatic Control*, 54(11):2692, 2009.

[24] Manuel A Duarte and Kumpati S Narendra. Combined direct and indirect approach to adaptive control. *Automatic Control, IEEE Transactions on*, 34(10):1071–1075, 1989.

[25] Girish Chowdhary and Eric Johnson. Concurrent learning for convergence in adaptive control without persistency of excitation. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3674–3679. IEEE, 2010.

[26] Girish Chowdhary, Tansel Yucelen, Maximillian Mühlegg, and Eric N Johnson. Concurrent learning adaptive control of linear systems with exponentially convergent bounds. *International Journal of Adaptive Control and Signal Processing*, 27(4):280–301, 2013.

[27] Girish Chowdhary, Maximilian Mühlegg, Jonathan P. How, and Florian Holzapfel. Concurrent learning adaptive model predictive control. In Qiping Chu, Bob Mulder, Daniel Choukroun, Erik-Jan Kampen, Coen Visser, and Gertjan Looye, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 29–47. Springer Berlin Heidelberg, 2013.

[28] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[29] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[30] G. Chowdhary, H.A. Kingravi, J.P. How, and P.A. Vela. Bayesian nonparametric adaptive control using gaussian processes. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–1, 2014.

[31] Robert C Grande, Thomas J Walsh, Girish Chowdhary, Sarah Ferguson, and Jonathan P How. Online regression for data with changepoints using gaussian processes and reusable models. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[32] Jay H Lee and Brian Cooley. Recent advances in model predictive control and other related areas. In *AIChE Symposium Series*, volume 93, pages 201–216. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997.

[33] S Joe Qin and Thomas A Badgwell. An overview of industrial model predictive control technology. In *AIChE Symposium Series*, volume 93, pages 232–256. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997.

[34] P. Bouffard, A. Aswani, and C. Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, page 279284, may 2012.

[35] Girish Chowdhary, Maximillian Muhlegg, Jonathan P How, and Florian Holzapfel. A concurrent learning adaptive-optimal control architecture for nonlinear systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 868–873. IEEE, 2013.

[36] Vilfredo Pareto. The new theories of economics. *Journal of Political Economy*, 5(4):485502, Sep. 1897.

[37] Nhan T Nguyen. Multi-objective optimal control modification adaptive control method for systems with input and unmatched uncertainties. In *AIAA Guidance, Navigation, and Control Conference*, page 0454, 2014.

[38] H-D Joos. A methodology for multi-objective design assessment and flight control synthesis tuning. *Aerospace Science and Technology*, 3(3):161–176, 1999.

[39] Nhan Nguyen and Ezra Tal. A multi-objective flight control approach for performance adaptive aeroelastic wing. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1843, 2015.

[40] WH Sadid, SL Ricker, and Shahin Hashtrudi-Zad. Multiobjective optimization in control with communication for decentralized discrete-event systems. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 372–377. IEEE, 2011.

[41] Adrian Gambier. Control of a reverse osmosis plant by using a robust pid design based on multi-objective optimization. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 7045–7050. IEEE, 2011.

[42] Mario Luca Fravolini, Tansel Yucelen, Jonathan Muse, and Paolo Valigi. Analysis and design of adaptive control systems with unmodeled input dynamics via multiobjective convex optimization. *American Control Conference (ACC)*, pages 1579–1584, 2015.

[43] Sean Shan-Min Swei, Guoming G Zhu, and Nhan Nguyen. Lmi-based multiobjective optimization and control of flexible aircraft using vcctef. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. AIAA SciTech, 2015.

[44] Rakshit Allamaraju, Hassan Kingravi, Allan Axelrod, Girish Chowdhary, Robert Grande, Christopher Crick, Weihua Sheng, and Jonathan How. Human aware path planning in urban environments with nonstationary mdps. In *International Conference on Robotics and Automation*, Hong Kong, China, 2014. IEEE.

[45] Joshua Joseph, Finale Doshi-Velez, A. S. Huang, and N. Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011.

[46] Girish Chowdhary, Hassan A Kingravi, Jonathan P How, and Patricio A Vela. Bayesian nonparametric adaptive control of time-varying systems using gaussian processes. In *American Control Conference (ACC), 2013*, pages 2655–2661. IEEE, 2013.

[47] Elmer G Gilbert, Ilya Kolmanovsky, and Kok Tin Tan. Nonlinear control of discrete-time linear systems with state and control constraints: A reference governor with global convergence properties. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 1, pages 144–149. IEEE, 1994.

[48] Francis Ysidro Edgeworth. *Mathematical psychics: An essay on the application of mathematics to the moral sciences*. C. Keagann Paul, 1881.

[49] Joseph Persky. Retrospectives: Pareto's law. *The Journal of Economic Perspectives*, pages 181–192, 1992.

[50] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, DTIC Document, 1999.

[51] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[52] Abraham Charnes and William W Cooper. Nonlinear power of adjacent extreme point methods in linear programming. *Econometrica: journal of the Econometric Society*, pages 132–153, 1957.

[53] FW Gembicki. *Vector optimization for control with performance and parameter sensitivity indices*. PhD thesis, Ph. D. Thesis, Case Western Reserve Univ., Cleveland, Ohio, 1974.

[54] Antonio López Jaimes, Saúl Zapotecas Martınez, and Carlos A Coello Coello. An introduction to multiobjective optimization techniques. *Optimization in Polymer Processing*, pages 29–57, 2009.

[55] Tao Zheng, Gang Wu, Guang-Hong Liu, and Qing Ling. *Multi-objective nonlinear model predictive control: Lexicographic method*. INTECH Open Access Publisher, 2010.

[56] Arthur M Geoffrion, James S Dyer, and A Feinberg. An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management science*, 19(4-part-1):357–368, 1972.

[57] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[58] Andrzej P Wierzbicki. *A methodological guide to multiobjective optimization*. Springer, 1980.

[59] Andrzej Jaszkiewicz and Roman Słowiński. The light beam searchapproach–an overview of methodology applications. *European Journal of Operational Research*, 113(2):300–314, 1999.

[60] Ralph E Steuer and Eng-Ung Choo. An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical programming*, 26(3):326–344, 1983.

[61] Saul Gass and Thomas Saaty. The computational algorithm for the parametric objective function. *Naval research logistics quarterly*, 2(1-2):39–45, 1955.

[62] Yacov Y Haimes, Leon S Lasdon, and David A Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems Man and Cybernetics*, (1):296–297, 1971.

[63] Milan Zeleny and James L Cochrane. *Multiple criteria decision making*. University of South Carolina Press, 1973.

[64] Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[65] Sauleh Siddiqui, Shapour Azarm, and Steven A Gabriel. On improving normal boundary intersection method for generation of pareto frontier. *Structural and Multidisciplinary Optimization*, 46(6):839–852, 2012.

[66] Giorgio Chiandussi, Marco Codegone, S Ferrero, and Federico Erminio Varesio. Comparison of multi-objective optimization methodologies for engineering applications. *Computers & Mathematics with Applications*, 63(5):912–942, 2012.

[67] Liuping Wang. *Model predictive control system design and implementation using MATLAB®*. springer, 2009.

[68] Pu Li, Moritz Wendt, and Günter Wozny. Robust model predictive control under chance constraints. *Computers & Chemical Engineering*, 24(2):829–834, 2000.

[69] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007.

[70] Eugene Lavretsky and Kevin Wise. *Robust and Adaptive Control: With Aerospace Applications*. Springer Science & Business Media, 2012.

[71] Girish Vinayak Chowdhary, Emilio Frazzoli, Jonathan P How, and Hugh Liu. Nonlinear flight control techniques for unmanned aerial vehicles. In *Handbook of Unmanned Aerial Vehicles*, pages 577–612. Springer, 2014.

[72] Robert M Sanner and J-JE Slotine. Gaussian networks for direct adaptive control. *Neural Networks, IEEE Transactions on*, 3(6):837–863, 1992.

[73] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[74] R.M. Sanner and J.-J.E. Slotine. Gaussian networks for direct adaptive control. *Neural Networks, IEEE Transactions on*, 3(6):837863, nov 1992.

[75] Morgan M. Monahemi and Miroslav Krstic. Control of wingrock motion using adaptive feedback linearization. *Journal of Guidance Control and Dynamics*, 19(4):905–912, August 1996.

APPENDIX A

Acronyms

Acronym	Expanded Version
GP	Gaussian Process
BNP	Bayesian Non-Parametric
MRAC	Model Reference Adaptive Control
MPC	Model Predictive Control
NBC	Non-Bayesian Clustering
RBF	Radial Basis Function
RKHS	Reproducing Kernel Hilbert Spaces

VITA

Ali Abdollahi

Candidate for the Degree of

MASTER OF SCIENCE

Thesis: ADAPTIVE MULTI-OBJECTIVE OPTIMIZING FLIGHT CONTROLLER

Major Field: Mechanical and Aerospace Engineering

Biographical:

Education: Completed the requirements for the Master of Science degree with a major in Mechanical and Aerospace Engineering at Oklahoma State University in April, 2016.

Experience: Research Assistant in Distributed Autonomous Systems Lab (DASLab) at Oklahoma State University, Stillwater, from Spring 2014 to present.

Teaching/Lab Assistant for Measurements and Instrumentation course at Oklahoma State University, Tulsa, Spring 2015.

Teaching Assistant for Engineering Analysis course at Oklahoma State University, Stillwater, Fall 2014.

Professional Memberships: AIAA and IEEE